University of California
Santa Barbara

Empowering Responsible Use of Large Language Models

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Xuandong Zhao

Committee in charge:

Professor Yu-Xiang Wang, Co-Chair
Professor Lei Li, Co-Chair
Professor Prabhanjan Ananth
Professor Dawn Song

June 2024

The Dissertation of Xuandong Zhao is approved.

Professor Prabhanjan Ananth

Professor Dawn Song

Professor Lei Li, Committee Co-Chair

Professor Yu-Xiang Wang, Committee Co-Chair

June 2024

Empowering Responsible Use of Large Language Models

by

Xuandong Zhao

To my family.

Acknowledgements

The past five years at UC Santa Barbara have been an unforgettable and invaluable experience for me. In 2019, after my first quarter as a PhD student, just as I was becoming familiar with the school and my research, the Covid-19 pandemic broke out globally. During this dark time, everyone was anxious and uncertain about the future, and studying online posed significant challenges to research progress. Fortunately, we persevered through this difficult period and entered a post-pandemic era. In 2022, nearly concurrent with the end of Covid-19, Large Language Models (LLMs) began to rapidly gain popularity and soon dominated almost all NLP applications. I was fortunate to witness the development of LLMs from the beginning and felt excited to be a part of this trend, especially regarding the responsible use of generative AI. I would not have been able to make this journey without the help and support of many people, and I feel deeply indebted to them.

I would like to express my deepest gratitude to my exceptional advisors, Yu-Xiang Wang and Lei Li, for their unwavering support, guidance, and mentorship throughout my PhD journey. They are, without a doubt, among the best advisors a PhD student could hope for.

Yu-Xiang, a respected computer scientist and statistician, has been an invaluable source of wisdom and inspiration to me. I learned from him that I should consistently uphold high standards in every project I undertake and strive to produce impactful research. During the early stages of my PhD, I faced challenges in understanding and making progress in theoretical research. Yu-Xiang's patience, care, and support were invaluable in helping me overcome these obstacles. He encouraged me to explore my research interests with complete freedom, offering guidance and vision that demonstrated his deep understanding of the field. His keen theoretical insight and meticulous pursuit

of mathematical rigor have been instrumental in leading numerous research projects to successful outcomes.

Lei is an enthusiastic researcher and selfless educator who possesses an acute instinct for identifying key research problems. Working with Lei has been an enriching experience; I always feel my passion ignited after talking to him. He worked alongside us late into the night before conference deadlines. Lei consistently encouraged me to share my research with the public and hone my writing and presentation skills. I am thankful for the freedom Lei granted me to explore diverse research interests and for connecting me with leading experts. From Lei, I have learned the importance of optimism, perseverance, and the steps to becoming an independent researcher.

I feel fortunate to study at the intersection of Yu-Xiang and Lei's research interests. Their supervision prepared me to be a researcher who thinks both theoretically and empirically. Their support and vision helped me push forward in groundbreaking research projects on AI security and privacy. With large language models being the revolution of the new AI field, I am excited that our research has made a real impact on the world.

I appreciate Prabhanjan Ananth and Dawn Song for serving on my thesis committee. It is an honor to have them on my committee. Their unique insight from different perspectives, feedback, and comments have made my work more solid. Their support and advice for my professional career are invaluable.

Many great minds have guided me and led me through this journey. I would not have become who I am today without the help of my undergraduate research mentors, Xi Li, Quanzheng Li, and Xiang Li, who introduced me to the exciting field of AI and supported me in pursuing a PhD overseas. I was fortunate enough to work with Zhiguo Yu and Ming Wu during internships at Microsoft. I also thank Sai Teja Peddinti, Benoit Seguin, and Nina Taft for their kindness and mentorship during my internship at Google. I look forward to future collaborations with them.

Throughout the years, I have had the pleasure of collaborating with many brilliant researchers. This thesis would not have been possible without the collaborative efforts of my esteemed colleagues: Dheeraj Baby, Siqi Ouyang, Yuqing Zhu, Chuan Guo, Xianjun Yang, Liangming Pan, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Wenda Xu, Guanglei Zhu, Jiayi Fu, Jiangjie Chen, André Vicente Duarte, Arlindo L. Oliveira, Weixin Liang, Zachary Izzo, Yaohui Zhang, Zhengxuan Wu, Tianyu Pang, Chao Du, Brian Tufts, Chenwen Liao, Aiwei Liu, Leyi Pan, Tong Zhou and Jielin Qiu. I have learned a great deal from each paper we wrote together, and I truly enjoyed working with each of them.

I would also like to thank other faculty members with whom I had memorable and helpful discussions about my work and career choices. My gratitude goes to William Yang Wang, Fei Fang, Xin Wang, Chong Liu, Wenbo Guo, Yao Qin, Xifeng Yan, Christopher Kruegel, Giovanni Vigna, Wenhu Chen, Zhiyu Chen, Yanghua Xiao, and Peng Zhao.

I have enjoyed the privilege of being part of the Machine Learning Group and Natural Language Processing Group. I am grateful to my friends and colleagues, including but not limited to Kaiqi Zhang, Ming Yin, Rachel Redberg, Jianyu Xu, Dan Qiao, Esha Singh, Erchi Wang, Momin Haider, Sunil Madhow, Danqing Wang, Zhenqiao Song, Tsu-Jui Fu, Weixi Feng, Jiabo Ji, Hong Wang, Yue Wei, Fuheng Zhao, Lianke Qin, Guyue Huang, Shiyang Li, Xinlu Zhang, Xiyou Zhou, Wanrong Zhu, Zekun Li, Weizhi Wang, and Yifan Qiao.

This thesis is affectionately dedicated to my beloved wife, Yang Gao, whose unwavering support and companionship have been my greatest strength throughout this journey. Her insightful feedback, continuous encouragement, and endless patience have not only propelled my academic endeavors but also deeply enriched every aspect of my life. I extend my heartfelt gratitude to my family - my loving parents and my caring sister - who have instilled in me the values of perseverance and dedication. I am equally grateful

to my wife's family, whose warmth, encouragement, and support have been sources of comfort and motivation. With the recent arrival of our first child, Henry, this milestone is not only a reflection of my personal commitment but also a tribute to the collective love, support, and sacrifices of our families, to whom I owe an immeasurable debt of gratitude.

Curriculum Vitæ

Xuandong Zhao

Education

2024	Ph.D. in Computer Science, University of California, Santa Barbara
2019	B.E. in Computer Science, Zhejiang University

Publications

[16] **Permute-and-Flip: An Optimally Robust and Watermarkable Decoder for LLMs**
Xuandong Zhao, Lei Li, Yu-Xiang Wang
arXiv, 2024

[15] **Weak-to-Strong Jailbreaking for Large Language Models**
Xuandong Zhao*, Xianjun Yang*, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, William Wang
arXiv, 2024

[14] **GumbelSoft: Diversified Language Model Watermarking via the GumbelMax-Trick**
Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, Yanghua Xiao
Proceedings of ACL, 2024

[13] **Perils of Self-Feedback: Self-Bias Amplifies in Large Language Models**
Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, William Yang Wang
Proceedings of ACL, 2024

[12] **Monitoring AI-Modified Content at Scale: A Case Study on the Impact of ChatGPT on AI Conference Peer Reviews**
Weixin Liang*, Zachary Izzo*, Yaohui Zhang*, Haley Lepp, Hancheng Cao, Xuandong Zhao, Lingjiao Chen, Haotian Ye, Sheng Liu, Zhi Huang, Daniel McFarland, James Zou
Proceedings of ICML, 2024

[11] **DE-COP: Detecting Copyrighted Content in Language Models Training Data**
André Vicente Duarte, Xuandong Zhao, Arlindo L. Oliveira, Lei Li
Proceedings of ICML, 2024

[10] **Provable Robust Watermarking for AI-Generated Text**
Xuandong Zhao, Prabhanjan Ananth, Lei Li, Yu-Xiang Wang
Proceedings of ICLR, 2024

[9] **Invisible Image Watermarks Are Provably Removable Using Generative AI**
Xuandong Zhao*, Kexun Zhang*, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, Lei Li
ICML Workshop on Challenges in Deploying Generative AI, 2023

[8] **A Survey on Detection of LLMs-Generated Content**
Xianjun Yang, Liangming Pan, Xuandong Zhao, Haifeng Chen, Linda Petzold, William Yang Wang, Wei Cheng
arXiv, 2023

[7] **"Private Prediction Strikes Back!" Private Kernelized Nearest Neighbors with Individual Rényi Filter**
Yuqing Zhu, Xuandong Zhao, Chuan Guo, Yu-Xiang Wang
Proceedings of UAI, 2023 ***Spotlight***

[6] **Protecting Language Generation Models via Invisible Watermarking**
Xuandong Zhao, Yu-Xiang Wang, Lei Li
Proceedings of ICML, 2023

[5] **Pre-trained Language Models can be Fully Zero-Shot Learners**
Xuandong Zhao, Siqi Ouyang, Zhiguo Yu, Ming Wu, Lei Li
Proceedings of ACL, 2023 ***Oral***

[4] **Distillation-Resistant Watermarking for Model Protection in NLP**
Xuandong Zhao, Lei Li, Yu-Xiang Wang
Findings of EMNLP, 2022

[3] **Provably Confidential Language Modelling**
Xuandong Zhao, Lei Li, Yu-Xiang Wang
Proceedings of NAACL, 2022 ***Oral***

[2] **Compressing Sentence Representation for Semantic Retrieval via Homomorphic Projective Distillation**
Xuandong Zhao, Zhiguo Yu, Ming Wu, Lei Li
Findings of ACL, 2022

[1] **An Optimal Reduction of TV-Denoising to Adaptive Online Learning**
Dheeraj Baby, Xuandong Zhao, Yu-Xiang Wang
Proceedings of AISTATS, 2021

Abstract

Empowering Responsible Use of Large Language Models

by

Xuandong Zhao

The rapid advancement of powerful Large Language Models (LLMs), such as ChatGPT and Llama, has revolutionized the world by bringing new creative possibilities and enhancing productivity. However, these advancements also pose significant challenges and risks, including the potential for misuse in the form of fake news, academic dishonesty, intellectual property infringements, and privacy leaks. In response to these concerns, this thesis explores approaches to promoting the responsible use of LLMs from both theoretical and empirical perspectives.

Three key approaches are presented: (1) Detecting AI-generated Text via Watermarking: We propose a robust and high-quality watermarking method called Unigram-Watermark and introduce a rigorous theoretical framework to quantify the effectiveness and robustness of LLM watermarks. Furthermore, we propose PF-Watermark, which achieves the best balance of high detection accuracy and low perplexity. (2) Protecting the Intellectual Property of LLMs: We safeguard the intellectual property of LLMs through novel watermarking techniques designed to prevent model-stealing attacks in both text classification and text generation tasks. (3) Privacy-Preserving LLMs: We employ Confidential Redacted Training (CRT) to train and fine-tune language generation models while protecting sensitive information. In summary, we propose a suite of algorithms and solutions to address LLMs' trending safety, security, and privacy concerns. We hope our studies provide valuable insights for researchers to explore exciting future research solutions that promote responsible AI development and deployment.

Contents

Chapter 1

Introduction

1.1 Overview

The world stands on the brink of a technological revolution with the rapid emergence of powerful generative Artificial Intelligence (AI) models, particularly Large Language Models (LLMs). LLMs, such as ChatGPT [1] and Llama [2], enable remarkable creative capabilities like generating fluent essays, translating hundreds of languages, and producing efficient code. However, the power of LLMs also brings significant risks, including the creation of fake news, fraud, scams, the potential automation of disinformation campaigns, infringements on copyright and intellectual property, and significant privacy breaches. This underscores the urgent need for research and policies to ensure the safety, security, and privacy of these rapidly advancing technologies.

Responsible AI is a top priority for governments and society, as it is one of the most critical challenges to address. Last year, the White House issued an executive order promoting the safe, secure, and trustworthy development of Artificial Intelligence [3]. In Europe, leaders convened an AI Safety Summit [4]. Furthermore, over 200 universities and companies joined the AI Safety Institute to develop the next generation of AI models

aligned with human values to shape a better future [5].

To address the growing global concerns surrounding AI ethics, this thesis is rooted deeply in the realms of AI safety, security, and privacy from a technical standpoint. By advancing statistical machine learning techniques tailored for a deep understanding of natural language processing applications, this thesis promotes the responsible use of large language models from both theoretical and empirical perspectives. Specifically, we study three critical problems related to the responsible use of LLMs:

1. How to detect AI-generated text to prevent misuse.
2. How to protect the intellectual property of large language models to prevent theft.
3. How to prevent large language models from generating sensitive information.

To tackle these challenges, we developed novel watermarking, intellectual property protection, and differentially private training techniques that provide theoretical guarantees and empirical evidence with many current state-of-the-art models.

1.2 Watermarking LLM-Generated Text

In the first part, we explore the watermarking of LLM-generated text. The rapid advancement of large language models like ChatGPT has led to a proliferation of synthetic content generation across sectors such as media, cybersecurity, public discourse, and education. Detecting LLM-generated content is crucial, and watermarking LLMs' text output is one of the most promising approaches to addressing the safety challenges of LLM usage.

We propose two prominent solutions in this domain [6, 7]. In Unigram-Watermark, we formally define robustness for text watermarking of LLMs and propose a rigorous

theoretical framework to quantify performance drops, detection accuracy, and security against post-processing. The Unigram-Watermark method ensures that the watermarked LLM remains close to the original LLM, while the Type I/II errors in detection decrease exponentially as the suspect text length increases. Experiments on diverse LLMs and datasets demonstrate superior detection accuracy and robustness against attacks, promoting responsible LLM use.

The second work builds upon the Permute-and-Flip (PF) decoding method, further exploring its properties and applications in watermarking. We demonstrate the Pareto optimality of the PF decoding method in balancing robustness and perplexity, highlighting its advantages over traditional decoding methods like softmax sampling. Additionally, we propose a novel "PF Watermark", analogous to the Gumbel-Watermark, showcasing its effectiveness in detecting watermarked text while preserving the indistinguishability of the watermarked decoder from its non-watermarked counterpart.

1.3 Distillation Resistant Model Watermark

In the second part, we focus on protecting the intellectual property of generative AI [8, 9]. The proliferation of free or low-cost generative AI APIs has heightened concerns over intellectual property (IP) theft through model distillation. For example, the Alpaca team recently claimed to distill ChatGPT for just $600 in API calls, demonstrating the urgency of this issue. This raises a pertinent question: how can we prevent model-stealing attacks through distillation?

We seek to prevent such model-stealing attacks by developing novel watermarking techniques tailored for large language models. In Chapter 5, we propose Distillation-Resistant Watermarking (DRW) to protect NLP models from being stolen via distillation. DRW protects a model by injecting watermarks into the victim's prediction probabilities

corresponding to a secret key, which can be detected by probing a suspect model. From a theoretical perspective, we prove that a protected model still retains the original accuracy within a certain bound.

Progressing further, we adapt the principles of DRW to text generation models, the default setting for large language models. Our method, Ginsew, provides an innovative methodology to protect these models. By injecting secret watermarking signals into decoding steps, we can ascertain the origin of the model, offering a formidable defense against unauthorized distillation. Notably, our method is one of the first to watermark LLMs directly in the decoding steps. Overall, this line of work makes crucial strides toward securing AI systems against IP infringement via model distillation.

1.4 Privacy-Preserving LLMs

In the third part, we examine privacy-preserving LLMs [10]. LLMs pose serious privacy risks by making it easier to extract and exploit personal data. Moreover, large language models may inadvertently memorize sensitive information, such as Social Security numbers, contained in their massive training datasets, as it is infeasible to manually screen this data at scale. To address these challenges, we have developed effective techniques to train LLMs while preserving privacy.

Our proposed Confidentially Redacted Training (CRT) method trains language generation models while protecting confidential segments. Drawing inspiration from differential privacy, CRT can provably prevent unintended memorization by randomizing parts of the training process. Moreover, we demonstrate that redaction with an approximately correct screening policy amplifies the confidentiality guarantee. These methods help mitigate privacy risks in the initial model training phase.

1.5 Summary and Contributions

In summary, this thesis makes novel contributions to securing LLMs and their outputs through provably robust watermarking techniques for both generated text and the models themselves, as well as confidential training procedures. The proposed methods aim to enable the responsible development and deployment of LLMs while mitigating risks of misuse, intellectual property theft, and data leakage. By providing a solid foundation of theory and practice, this work helps chart a path towards a future where the immense benefits of LLMs can be realized while upholding important societal values.

1.5.1 Thesis Structure

The thesis is organized as follows:

- Chapter 1 introduces the background, motivation, and scope of the thesis.
- Chapter 2 presents the Unigram-Watermark method, offering a rigorous theoretical framework and empirical evidence for detecting LLM-generated text.
- Chapter 3 explores the PF Decoding method, delving into its robustness, perplexity trade-off, and applications in watermarking.
- Chapter 4 introduces Distillation-Resistant Watermarking (DRW), focusing on protecting NLP models from being stolen via distillation.
- Chapter 5 extends DRW principles to text generation models, introducing Ginsew for watermarking LLMs in decoding steps to prevent model stealing attacks.
- Chapter 6 discusses Confidentially Redacted Training (CRT), providing a new approach to training LLMs while preserving privacy.

- Chapter 7 concludes the thesis by summarizing key contributions and outlining future research directions.

This thesis contributes significantly to the responsible use of large language models, promoting safety, security, and privacy in a rapidly evolving technological landscape.

Part I

Watermark LLM-Generated Text

Chapter 2

Unigram-Watermark

We study the problem of watermarking large language models (LLMs) generated text — one of the most promising approaches for addressing the safety challenges of LLM usage. In this chapter, we propose a rigorous theoretical framework to quantify the effectiveness and robustness of LLM watermarks. We propose a robust and high-quality watermark method, Unigram-Watermark, by extending an existing approach with a simplified fixed grouping strategy. We prove that our watermark method enjoys guaranteed generation quality, correctness in watermark detection, and is robust against text editing and paraphrasing. Experiments on three varying LLMs and two datasets verify that our Unigram-Watermark achieves superior detection accuracy and comparable generation quality in perplexity, thus promoting the responsible use of LLMs. Our code is available at https://github.com/XuandongZhao/Unigram-Watermark.

2.1 Introduction

Generative Artificial Intelligence (AI) [11, 12, 13, 14] has achieved significant progress in recent years, spanning from computer vision (CV) to natural language processing

(NLP). Large language models (LLMs) such as ChatGPT [1] can generate coherent and contextually relevant long-form text in response to user-specified prompts. However, the ease of using LLMs has raised concerns about their potential misuse [15, 16, 17]. For example, LLMs could be used to generate fake news, contaminate web content, or assist in academic dishonesty. Additionally, the proliferation of synthetic data from LLMs poses challenges for training new models, as synthetic data needs to be detected and excluded before model training [18, 19].

There are two main camps of existing attempts to address these challenges. One camp, inspired by [20], aims at generically distinguishing machine-generated text from that of the humans [21, 22, 23, 15, 24]. These works primarily leverage hand-crafted or learned "statistical patterns" of generated text, thus their performance is not robust to distribution changes (e.g., by prompting / conditioning), prone to biases [25], and vulnerable to adversarial attacks.

The other camp advocates active intervention by injecting carefully-designed watermarks to machine-generated text [26, 9]. The watermarking approach does not search for statistical patterns (which could be hit-or-miss), but rather deliberately *plant* subtle but distinctive patterns within the content to enable downstream detection. Compared to the passive detection approaches, the watermarking methods aim at determining whether the text is coming from a *specific* language model rather than solving the Turing test generically. As a result, watermarking approaches are robust to distribution-shift and can essentially *prove* — rather than *predict* — the origin of the suspect text.

The most notable challenge for the watermarking approach is that the planted patterns could be post-processed away. As an example, [26]'s soft watermarking method divides the vocabulary into a "green list" and a "red list" based on the prefix token, and subtly increases the probability of choosing from the green list. If the watermarked sentence is edited by changing every other token into its synonym, then it is no longer

possible to determine the green/red lists for each candidate token, thus ruining the detector. One could also simply paraphrase the sentence as a whole using another off-the-shelf LLM.

In this chapter, we take a first stab at formally defining robustness in the context of watermarking LLMs. Our contributions are fourfold.

1. We devise a rigorous theoretical framework for quantifying the performance drop, the correctness of detection, and the security property against post-processing.

2. We propose to simplify the scheme of [26] by using a fixed Green-Red split consistently and show that the new watermark, named UNIGRAM-WATERMARK, is *twice as robust* to edits as the baseline, provably.

3. We prove that the watermarked LLM is close to the original LLM (in all Renyi divergences) and show that the Type I/Type II errors of the detection algorithm decay exponentially as the suspect text length gets longer and more diverse.

4. We conduct experiments utilizing various large language models on diverse datasets. The results indicate that our method achieves superior detection accuracy and improved robustness against different attacks, thus promoting the responsible use of LLMs.

To the best of our knowledge, we are the first to obtain provably robust guarantees for watermarks for LLMs against arbitrary edits.

Related work. We build upon the work of [26] in which the family of K-gram (statistical) watermark was proposed[1]. The main method we consider chooses $K = 1$, thus its name UNIGRAM-WATERMARK. Our work provides formal theoretical guarantees

[1]Note the changed name. [26] referred to its (unnamed) soft-watermark that determines the green/red list using a prefix of length $(K-1)$. We think K-gram watermark is the most concise and informative name for this family.

to this family of K-gram watermark. For the sake of a clean presentation, we focus on the case when $K = 1$ and discuss the applicability of our results for $K > 1$ in the discussion section. Our work is independent of the concurrent work of cryptographic watermarks [27, 28]. In particular, [27]'s proprietary work can also be viewed as an alternative K-gram watermark, but uses a cryptographic approach for measuring utility drop, which results in a different kind of tradeoff. We defer detailed discussion to an extended discussion of the related work in the next section. Technically, the main theoretical tool we used for analyzing dependent random variables and their concentration tightly is due to [29], the instantiation to our problem is new and nontrivial.

2.2 Related Work

Watermarking natural languages. The concept of watermarking, which involves hiding identifying information within data, has a long history. However, watermarking digital text has been challenging due to its discrete nature [30]. Early approaches relied on techniques such as synonym substitution [31], syntactic structure restructuring [32], or paraphrasing [33]. Later, advancements in modern neural language models led to improved methods that move away from rule-based approaches. Different approaches have been proposed, such as encoding messages by context-aware lexical substitution [34] or using mask-infilling models for editing text [35]. Recent studies [9, 26] explore modifying the logits of language models during token generation and embedding invisible watermarks in the decoding process. Our objective is to develop a robust watermarking technique for natural language models that maintain high text quality while effectively concealing identifying information.

Post-hoc detection. Rather than watermarking, an alternative approach involves developing detection models for post-hoc analysis of machine-generated text. Some de-

tection methods use statistical outlier detection techniques without requiring additional training. For example, GLTR [21] assesses the expected probability of individual tokens and applies thresholding to identify AI-generated content. DetectGPT [22] suggests that AI-generated passages tend to reside in the negative curvature of the log probability of texts. Another set of methods relies on classifiers that are fine-tuned to distinguish between human-written and machine-generated text. Initial efforts in this domain focus on detecting fake reviews [23] and fake news [15]. More recently, OpenAI releases a web interface that uses a finetuned GPT model for this discrimination task [24]. However, as language models improve, AI-generated text is becoming increasingly similar to human-generated text, making it more challenging to detect. [36] find that existing detection strategies designed for GPT-2 struggle with GPT-3. Moreover, known detectors are found to be fragile to adversarial attacks [37] and biased towards non-native English writers [25].

Impossibility results? [38] poses the question of whether detecting machine-generated text is possible and argue that as the human distribution and LLM distribution of texts get closer, any classifier will have to either have a large Type I error or a large Type II error. The authors also argue that (in Corollary 2) if the watermarking scheme can be learned then paraphrasing attacks either evade the detector or also classify humans with a similar distribution as false positives. This does not invalidate our results as we made no theoretical claim about paraphrasing. We do claim that in Theorem 2.3.4 that the watermarked LM $\hat{\mathcal{M}}$ and original LM $\mathcal{M}$ is statistically close — in fact, indistinguishable in the "differential privacy" sense. But the indistinguishability is for each token. As the number of tokens gets larger, they will eventually become distinguishable, that is why our Theorem 2.6.4 and Theorem 2.6.13 are not contradicting Theorem 2.3.4. This argument was initially pointed out by [39], showing that detection is possible.

Language model watermarks with provable guarantees. Concurrent to our

work, [28] consider the problem of formally defining watermarking language models and propose a construction with provable guarantees. The main differences between their work and ours are:

- In [28], the watermarked distribution is computationally indistinguishable (i.e., indistinguishable against probabilistic polynomial-time algorithms) from the un-watermarked distribution whereas in our case, we insist that the watermarked distribution is statistically close to the un-watermarked distribution (of each token). The Type-I/Type-II error guarantees and the security properties are qualitatively different in both works.

- We both use different approaches to achieve our definitions. The advantage of our construction is that it satisfies robustness to edits property whereas they have no such guarantees. On the other hand, our construction uses a very different set of assumptions (e.g., high entropy) on the language model and prompt that appears to be incompatible with theirs.

- Finally, we implement our construction and conduct a thorough empirical evaluation to demonstrate its practicality while they don't provide any implementation of their construction.

Statistical vs Cryptographic Watermarks. [28] and [27] are examples of *cryptographic* watermarks, while [26] and this work study *statistical* watermarks. There are several prominent differences that make it a bit challenging to compare the two kinds, but we will try. To start, we argue that both [28] and [27] use a similar definition of language model watermarks as Definition 2.3.2 and considered a similar set of properties. Specifically, the "soundness", "completeness" from [28] directly map to our "Type I error" and "Type II error" requirements. As we understand from the materials in [27]'s talk, their

"indistinguishability" is a form of performance guarantee for $\hat{\mathcal{M}}$. The difference to ours is that they require (in our notation)

$$\mathop{\mathbb{P}}_{\hat{\mathcal{M}}(\text{prompt})}[\text{Next token}] = \mathop{\mathbb{E}}_{\mathsf{k}}\left[\mathop{\mathbb{P}}_{\hat{\mathcal{M}}(\text{prompt})}[\text{Next token}|\mathsf{k}]\right] = \mathop{\mathbb{P}}_{\mathcal{M}(\text{prompt})}[\text{Next token}]$$

where the random key k is marginalized out. while our results require that for every k the next token

$$\mathop{\mathbb{P}}_{\hat{\mathcal{M}}(\text{prompt})}[\text{Next token}|\mathsf{k}] \approx_\delta \mathop{\mathbb{P}}_{\mathcal{M}(\text{prompt})}[\text{Next token}]$$

to be statistically close (in the same sense of δ-differential privacy). By our metric, however, [27]'s watermark does not appear to satisfy any nontrivial δ guarantee, since it only requires *unbiasedness*. For that reason, the detection guarantee and its tradeoff with quality that we discussed in Remark 2.6.15 is not applicable to the cryptographic watermarks.

2.3 Proposed Method: Unigram-Watermark

We start with an overview of the language model watermarking problem. The definitions and notations introduced in this section will be used throughout the chapter.

Language models. A language model (LM) $\mathcal{M}$ is a statistical model that describes the probability of a sequence of words occurring in a sentence. Common neural language models (e.g., GPT-2/3 [40, 11]) are designed for next-word prediction which typically uses a transformer neural network [41]. The LM has a "vocabulary" $\mathcal{V}$ with $N := |\mathcal{V}| = 50,000$ tokens or more [40, 42]. Let x be an input prompt. $y := [y_1, \ldots, y_n]$ are n tokens generated by $\mathcal{M}$. During inference, $\mathcal{M}$ receives the input prompt x as the prefix of generation. It iteratively computes logit scores $\boldsymbol{\ell}_t$ for every next token. The logits transform into a probability distribution via soft-(arg)max function $p_t[v] = \frac{\exp(\boldsymbol{\ell}_t[v])}{\sum_{i\in\mathcal{V}}\exp(\boldsymbol{\ell}_t[i])}$ for all $v \in \mathcal{V}$.

The LM then samples the next token from this distribution: $y_t \sim p_t$.

2.3.1 Definition of language model watermarking

In the language model watermarking problem, the objective for the model owner is to embed a secret message known as "watermark" within the generated sequence y for a given prompt x. There are two desired requirements for watermarking. First, the quality of the watermarked model should be comparable to the quality of the original, un-watermarked model. Second, an adversary needs to modify sufficiently many AI-generated text in order to evade detection.

Definition 2.3.1 (Edit distance)**.** The edit distance, denoted as $\mathsf{ED}(y, \boldsymbol{z})$, quantifies the number of basic operations required to transform a sequence y into another sequence $\boldsymbol{z}$. These operations include "insertion", "deletion", and "replacement" of tokens.

Definition 2.3.2 (Language model watermarking)**.** A **language model watermarking scheme** consists of two probabilistic polynomial-time algorithms $(\mathsf{Watermark}, \mathsf{Detect})$:

- $\mathsf{Watermark}(\mathcal{M})$: Let $\mathcal{M}$ be a language model and let $\mathbf{p}_t := \mathbb{P}_{\mathcal{M}(x)}[y_t = \cdot | y_{1:t-1}]$ be the *conditional* probability distribution of t-th token on $\mathcal{V}$ generated by $\mathcal{M}$. This algorithm produces a new model $\hat{\mathcal{M}}$ with a new conditional distribution $\hat{\mathbf{p}}_t := \mathbb{P}_{\hat{\mathcal{M}}(x)}[y_t = \cdot | y_{1:t-1}]$ on $\mathcal{V}$. Additionally, it outputs a detection key k associated with $\hat{\mathcal{M}}$. The watermark could contain certain randomness.

- $\mathsf{Detect}(\mathsf{k}, y)$: This algorithm takes input detection key k and sequence y, then outputs 1 (indicating it was generated by $\hat{\mathcal{M}}$) or 0 (indicating it was *not* generated by $\hat{\mathcal{M}}$).

We require the following **three correctness properties** to hold:

- ω-Quality of watermarked output, for $\omega \in \mathbb{R}$: Assume the original language model $\mathcal{M}$ generates a probability vector p_t for the token at position t. The watermarked model $\hat{\mathcal{M}}$

predicts the token at position t using the modified probability vector $\hat{p}_t$. It is required that the distance between the two probability distributions satisfies: $D\left(\hat{p}_t \| p_t\right) \leq \omega$ for any fixed prompts and prefixes.

- α_y-Type I error ("No false positives"): for any fixed y (i.e., independent to k), it holds that

$$\mathbb{P}\left[\mathsf{Detect}(\mathsf{k}, y) = 1 \ ; \ (\hat{\mathcal{M}}, \mathsf{k}) \sim \mathsf{Watermark}(\mathcal{M})\right] \leq \alpha_y.$$

- $\beta_{(x,\mathcal{M})}$-Type II error ("No false negatives"):

$$\mathbb{P}\left[\mathsf{Detect}(\mathsf{k}, y) = 0 \ ; \ \begin{smallmatrix}(\hat{\mathcal{M}},\mathsf{k})\sim\mathsf{Watermark}(\mathcal{M}) \\ y\sim\hat{\mathcal{M}}(x)\end{smallmatrix}\right] \leq \beta_{(x,\mathcal{M})}.$$

We also require the following **security property** (parameterized by $\epsilon \geq 0$ and $\eta(y, \mathsf{k}, \epsilon)$):

- For any adversary $\mathcal{A}$ that postprocesses y with auxiliary information aux and any prompt $x \in \mathcal{V}^*$

$$\mathbb{P}\left[\mathsf{Detect}(\mathsf{k}, y_{\mathcal{A}}) = 1 \text{ or } \mathsf{ED}(y, y_{\mathcal{A}}) \geq \eta(\mathsf{k}, y, \epsilon) \middle|_{\mathsf{Detect}(\mathsf{k},y)=1}^{y,\mathsf{k},} \ ; \ \begin{smallmatrix}(\hat{\mathcal{M}},\mathsf{k})\sim\mathsf{Watermark}(\mathcal{M}) \\ y\sim\hat{\mathcal{M}}(x) \\ y_{\mathcal{A}}\sim\mathcal{A}(y,\mathsf{aux})\end{smallmatrix}\right] \geq 1 - \epsilon.$$

Remark 2.3.3 (Discussion on Definition 2.3.2). Informally, our definition allows us to formally quantify the essential properties of a language model watermarking scheme including its generation quality relative to the input LM, the accuracy of detection in terms of both false positives and false negatives, as well as the robustness to attacks.

The security property, in particular, states the following: suppose a malicious adversary intends to evade the detection algorithm, then the adversarial answer, to some input prompt x, should be far away (in edit distance) from any AI-generated answer. In other words, the optimal strategy to evade the detection algorithm would necessitate

executing a minimum number of insert/delete/replacement operations, captured by the function $\eta(\cdot)$ in Definition 2.3.2. This conceptually suggests that the adversary must exert considerable effort to successfully elude detection.

Admittedly, there are other attacks where edit distance does not capture either the effort or the utility loss. For example, if one prompts an unwatermarked LLM to paraphrase y then the number of edits can be large but the semantic meaning is retained. However, edit distance is a natural metric that smoothly interpolates the gray zone between the world where $y_{\mathcal{A}} = y$ in which it should clearly be caught and the other world where $y_{\mathcal{A}}$ is *independently created* without using $\hat{\mathcal{M}}$ in which it would be a false positive if Detect returns 1.

2.3.2 Threat models

Adversary's objective. The primary objective of the adversary is to render the watermark detection algorithm ineffective. Specifically, the adversary aims to produce a $y_{\mathcal{A}}$ such that $\mathsf{Detect}(\mathsf{k}, y_{\mathcal{A}}) = 0$ while at the same time, $y_{\mathcal{A}}$ is a minor modification of an AI-generated text y.

Adversary's capabilities. We consider an adversary with black-box input-output access to the language model. This adversary has the capacity to modify the sequence within a *bounded edit distance*. Given an input prompt x, the watermarked language model generates a text output $y \leftarrow \hat{\mathcal{M}}(x)$. The adversary, equipped with arbitrary side-information and computational resources, can then produce a modified output $y_{\mathcal{A}}$ such that the edit distance between the original and modified output, $\mathsf{ED}(y, y_{\mathcal{A}})$, is bounded, i.e. $\mathsf{ED}(y, y_{\mathcal{A}}) < \eta$.

2.3.3 Method

Algorithm 1 UNIGRAM-WATERMARK: Watermark

Input: random number generator F, green list size $\gamma \in (0, 1)$, watermark strength δ.
Randomly generate a watermark key k using F.
Use watermark key to partition the vocabulary of $\mathcal{M}$ into a "green list" $G \subset \mathcal{V}$ of size $\gamma|\mathcal{V}|$, and a "red list" $R = G^c$.
Define a new language model $\hat{\mathcal{M}}$ where for t and any prefix $[x, y_{1:t-1}]$, the resulting logits satisfy

$$\hat{\boldsymbol{\ell}}_t[v] := \boldsymbol{\ell}_t[v] + \delta \mathbf{1}(v \in G), \tag{2.1}$$

where $\mathbf{1}(\cdot)$ is the indicator function and the logit vector $\boldsymbol{\ell}_t \in \mathbb{R}^{|\mathcal{V}|}$ is obtained by the passing the same prefix to $\mathcal{M}$.
Output: watermark key k, watermarked language model $\hat{\mathcal{M}}$.

Algorithm 2 UNIGRAM-WATERMARK: Detect

Input: suspect text y, watermark detection key k, threshold τ.
Output: 1 or 0 (whether the text is watermarked).
Use the watermark detection key k to find the "green list" G.
Calculate the number of green list tokens $|y|_G = \sum_{t=1}^{n} \mathbf{1}(y_t \in G)$ in $[y_1, \ldots, y_n]$.
Compute the z-statistic:

$$z_y = \left(|y|_G - \gamma n\right) / \sqrt{n\gamma(1-\gamma)}. \tag{2.2}$$

if $z_y > \tau$ **then return** 1, i.e., "The suspect text is watermarked."
else return 0, i.e., "The suspect text is not watermarked."

Now let us instantiate Definition 2.3.2 with concrete algorithms. We will focus on UNIGRAM-WATERMARK — a variant of the K-gram watermark proposed by [26] but with a choice of $K = 1$. Pseudocodes of our approach Watermark and Detect are provided in Algorithm 1 and 2.

In Algorithm 1, we randomly partition the vocabulary into two distinct sets: the green list with γN tokens and the red list with the remaining tokens. In $\hat{\mathcal{M}}$, the logits of the language model for the green list tokens are increased by δ while the logits for tokens in the red list remain unchanged. Then at detection time (Algorithm 2), we count the number of green tokens in the suspect text, normalize the test-statistic, then make

a calibrated decision on whether we think the suspect text is generated from $\hat{\mathcal{M}}$ or not. We show the examples of real prompts and watermarked outputs in Table 2.1.

The watermarking procedure is parameterized by two *watermark strength parameters* γ, δ. γ determines the fraction of the vocabulary included in the green list. We typically set γ to be a constant, e.g., $1/3$ or 0.5. δ specifies the increase in the logits associated with the green list tokens. The larger δ is, the lower the quality of the watermarked LM, but the easier it is to detect.

Our UNIGRAM-WATERMARK enjoys all good properties of the general K-gram watermark from [26]. It runs in linear time and does not require access to the language model or the prompt used for generation. It is also intuitively robust to cropping and minor edits.

Overall, the proposed watermarking scheme requires almost no overhead in its implementation, is extremely simple, and is easy to maintain. The big question is:

`How well does this watermark scheme work?`

The remainder of this chapter provides answers to this question with provable guarantees (Section 2.3.3) on the properties from Definition 2.3.2 and extensive experiments (Section 2.4).

Before that, let us address two burning questions that a knowledgeable reader may have.

Why choosing $K = 1$? Recall that the general K-gram watermark works in the same way as ours, but randomly generates a different Green list for each prefix of length $K - 1$. In contrast, choosing $K = 1$ means we have a consistent green list for every new token the language model generates. The main advantage of choosing $K = 1$ is that it is the most robust choice within this family — and we believe robustness is the single most important feature of a watermarking scheme in practice.

Prompt	Is it possible to fail several times and succeed once?
LLaMA-13B (un-watermarked) PPL: 12.19 z-score: -2.425	Of course it is, and that is how we improve.\nSaying "I can't do that" is never a good thing.\nSometimes we think we've tried all we can and that "isn't enough". That is the time when we ask for help.\nThe root of all evils is to be a secret. Honesty and self-criticism is necessary for improvement. \nThe measure of intelligence is the ability to change. [continues...]
LLaMA-13B (watermarked) PPL: 12.47 z-score: 11.085	When most people are confronted with failure, they cannot imagine such a thing happening. When one faces business reverses and bankruptcy, it seems impossible. When we are rejected it looks as if we are going to be rejected forever. However, it does not need to be this way. The human spirit simply will not give up. When we face failure and it looks like we [continues...]

Table 2.1: Comparison of un-watermarked and watermarked text using the LLaMA-13B model. Green and red tokens are color-coded respectively. UNIGRAM-WATERMARK produces watermarked text of similar quality without noticeable degradation, yet with significant differences in z-scores for watermark detection.

Robustness to other attacks. Besides the robustness to edits, which we will prove in Section 2.3.3 and compare to that of $K \geq 2$. UNIGRAM-WATERMARK is also resilient to many other kinds of generation time attacks that people can apply such as reversing, shuffling, as well as the "Emoji insertion attack" that will completely break the watermark for $K \geq 2$ but not for $K = 1$. We provide a detailed discussion of this in Appendix 2.6.10.

The price for robustness? [26] did not consider the choice of $K = 1$ for an obvious reason. The watermark is now so simple that an attacker who observes the generated text may learn to guess the consistent green list. This is an issue for $K \geq 2$ too but certainly more so for $K = 1$. There is a robustness-learnability tradeoff as we adjust K which deserves a more rigorous treatment in future work. That said, we are ready to argue for biasing towards robustness. Why? We argue that in practice, it could be surprisingly difficult for an attacker to construct a meaningful attack when they do not have access to the original LM. We provide a more detailed experimental study with a faithful practical attack in Appendix 2.5.4. Moreover, there are alternative ways to get around this issue by refreshing the green list once in a while.

Main theoretical results

In this section, we present the quality, correctness, and security properties of UNIGRAM-WATERMARK as described in Definition 2.3.2.

2.3.4 Quality guarantee of Unigram-Watermark

We first show that the distance between the original probability vector p_t and the watermarked probability vector $\hat{p}_t$ are very close to each other in any Renyi-divergence.

Theorem 2.3.4. *Consider h as the input to the language model at step t, denoted as $h = [x, y_{1:t-1}]$. Fix green list G. Let δ represent the watermark strength. For any h, the α-th order Renyi-divergence between the watermarked probability distribution $\hat{p}_t = \hat{p}_t(\cdot|h)$ at time step t and the original probability distribution $p_t = p_t(\cdot|h)$ satisfies:*

$$\forall h, \max\left(D_\alpha\big(\hat{p}_t \| p_t\big), D_\alpha\big(p_t \| \hat{p}_t\big)\right) \leq \min\{\delta, \alpha\delta^2/8\}.$$

The proof, deferred to the appendix, leverages a surprising connection to modern techniques in the differential privacy literature [43, 44].

Remark 2.3.5 (KL-divergence and other probability distance metrics). Renyi-divergence is very general. Kullback-Leibler-divergence and chi-square divergence are directly implied by the α-Renyi divergence bound of $\min\{\delta, \alpha\delta^2/8\}$ by choosing $\alpha = 1$ and $\alpha = 2$ respectively and swap $\hat{p}$ and p. Hellinger distance can be obtained by choosing $\alpha = 0.5$. By Pinsker's inequality, we get a Total Variation distance bound of $\min\{\sqrt{\delta/2}, \delta/4\}$. Moreover, by choosing $\alpha \to \infty$, we obtain an upper bound of δ for a very strong multiplicative guarantee known as max-divergence. The resulting two distributions $\hat{p}$ and p are referred to by cryptographers as $(\delta, 0)$-*indistinguishable*, which says that for any measurable event S, the log-odds ratio satisfies $-\delta \leq \log \frac{\hat{p}_t(y_t \in S|h)}{p_t(y_t \in S|h)} \leq \delta$.

To summarize, our result shows that Algorithm 1 produces $\hat{\mathcal{M}}$ that satisfies ω-quality of watermarked output with ω (as a function of δ) for almost all commonly used probability distance D.

2.3.5 Type I error of Unigram-Watermark

Theorem 2.3.6 (No false positives (short version of Theorem 2.6.4))**.** *Consider $y = y_{1:n}$ as any* fixed *text. Define $C_{\max}(y) := \max_{i\in[N]} \sum_{j=1}^{n} \mathbf{1}(y_j = i)$ and $V(y) := \frac{1}{n}\sum_{i=1}^{N}(\sum_{j=1}^{n} \mathbf{1}(y_j = i))^2$. With probability $1-\alpha$ (over only the randomness of G):*

$$z_y \leq \sqrt{\frac{64V(y)\log(9/\alpha)}{1-\gamma}} + \frac{16C_{\max}(y)\log(9/\alpha)}{\sqrt{n\gamma(1-\gamma)}}.$$

The theorem says that the z-score for any sufficiently diverse text is $\tilde{O}(1)$ and it is applicable to any text not generated by the watermarked LM $\hat{\mathcal{M}}$.

Remark 2.3.7 (Controlling false positive rate). The theorem implies that if we choose $\tau > \sqrt{\frac{64V\log(9/\alpha)}{1-\gamma}} + \frac{16C_{\max}\log(9/\alpha)}{\sqrt{n\gamma(1-\gamma)}}$, then the false-positive rate is smaller than α. Note that V and $C_{\max}$ can be computed directly from y, allowing us to choose an input-dependent τ as a function of $V, C_{\max}$ that achieves a α-Type I error guarantee with a fixed α for all inputs. In particular, the Type I error α decreases exponentially as we increase the threshold τ.

2.3.6 Type II error of Unigram-Watermark

To bound the Type II error, i.e., false negative rates, we need to make certain assumptions about p of the language model and the prompt x. These assumptions include a **"on-average high entropy"** assumption and a **"homophily"** condition. We will provide a detailed definition and discussion of these assumptions in Appendix 2.6.4 and

Appendix 2.6.4.

The "on-average high entropy" assumption requires the probability of the roll-out text to be "sufficiently diverse" on average. It is related but different from the "spike entropy" assumption used by [26]. The "homophily" assumption is new to this chapter. It is an assumption about the distribution induced by the state-transitions of the language model $\mathcal{M}$, which says that increasing the probability of a green-list token at time t does not decrease the probability of seeing that token in the future. This may seem counter-intuitive, but we will give concrete examples in Appendix 2.6.4 to show why this is fundamental for any statistical watermark to work effectively.

Theorem 2.3.8 (Only true positive (informal version of Theorem 2.6.13))**.** *Assume "average-high entropy" and "homophilly" to be valid with appropriate parameters, and in addition* $n \geq \tilde{\Omega}(\log(1/\beta)/\delta^2)$*, then with probability* $1-\beta$*,*

$$z_y \geq \Omega\left((e^{\delta}-1)\sqrt{n\gamma(1-\gamma)}\right).$$

Remark 2.3.9. The bounds on Type I/II error together say that $z_y \asymp \delta\sqrt{n}$ if y is from $\hat{\mathcal{M}}$ while $z_y \asymp O(1)$ otherwise, i.e., there is a large margin between them so we can choose τ in between. Also, the α and β parameters decay exponentially as the n gets larger.

2.3.7 Security property of Unigram-Watermark

We demonstrate the robustness of our watermarking scheme against editing attempts through Theorem 2.3.10. As a baseline of comparison, we also obtain new robustness guarantees for the soft watermarking method proposed in [26]. The detailed proof is deferred to the Appendix 2.6.

Theorem 2.3.10 (Robustness to editing)**.** *Let* $y = [y_1, \ldots, y_n]$ *represent the watermarked*

sequence. Suppose the adversary $\mathcal{A}$ follows Definition 2.3.2 and outputs a modified text $u = [u_1, \ldots, u_m]$. Following Equation 2.2, we calculate z-score z_y and z_u. Assume edit distance between y and u (denoted as η) satisfies $\eta < n$. Then we have

$$z_u \geq z_y - \max\left\{\frac{(1+\gamma/2)\eta}{\sqrt{n}}, \frac{(1-\gamma/2)\eta}{\sqrt{n-\eta}}\right\}.$$

In particular, when $\eta \leq \frac{2\gamma n}{(1+\gamma/2)^2}$, we can drop the second term in the max.

This theorem bounds the changes to our test z-score when η edits are performed. As we established for a high-entropy sequence, z_y typically grows in $O((e^\delta - 1)\sqrt{n})$, which means that when δ is a constant, with an appropriate choice of τ, the watermark is robust up to $O(n)$ arbitrary edits! Finally, compared to [26]'s watermark, ours is twice as robust (see Appendix 2.6.7).

2.4 Experiments

In this section, we aim to conduct experiments to evaluate watermark detection performance, watermarked text quality, and robustness against attacks compared to the baseline. Additional experiment results including different parameters, white-box attacks, scaled language models, etc. are deferred to Appendix 2.5.

2.4.1 Experiment setting

Datasets and prompts. We utilize two long-form text datasets: OpenGen and LFQA. OpenGen, collected by [45], consists of 3K two-sentence chunks sampled from the validation split of WikiText-103 [46]. The subsequent 300 tokens serve as the human-written continuation. LFQA is a long-form question-answering dataset created by [45] by scraping questions from Reddit, posted between July and December 2021, across

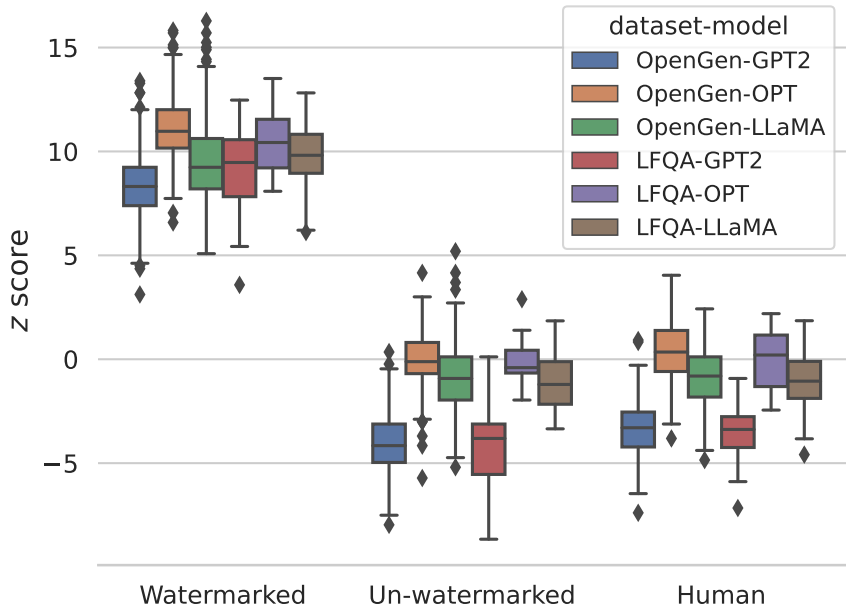

Figure 2.1: z-scores of watermarked and un-watermarked machine-generated text, along with the z-score of human-generated text. The watermarked text z-score surpasses the empirical threshold of $z = 6.0$.

six domains. [45] randomly select 500 questions from each domain and pair them with their corresponding longest human-written answers, resulting in 3K QA pairs. In our experiments, we use the questions as prompts and the corresponding answers as human-written text.

Language models. We conduct experiments using three state-of-the-art public language models of varying sizes from different model families: GPT2-XL with 1.5B parameters [40], OPT-1.3B [47], and LLaMA-7B [48]. Nucleus Sampling [49] is employed as the default decoding algorithm to introduce randomness while maintaining human-like text output. The models are loaded from the Huggingface library [50], and the `generate` API function is used to adjust the logits distribution of the language model.

Evaluation methods. Maintaining a low false positive rate is crucial to prevent misclassifying un-watermarked text as watermarked. To ensure this, we set the false positive rates at 1% and 10% for all detection algorithms and adjust the detection threshold

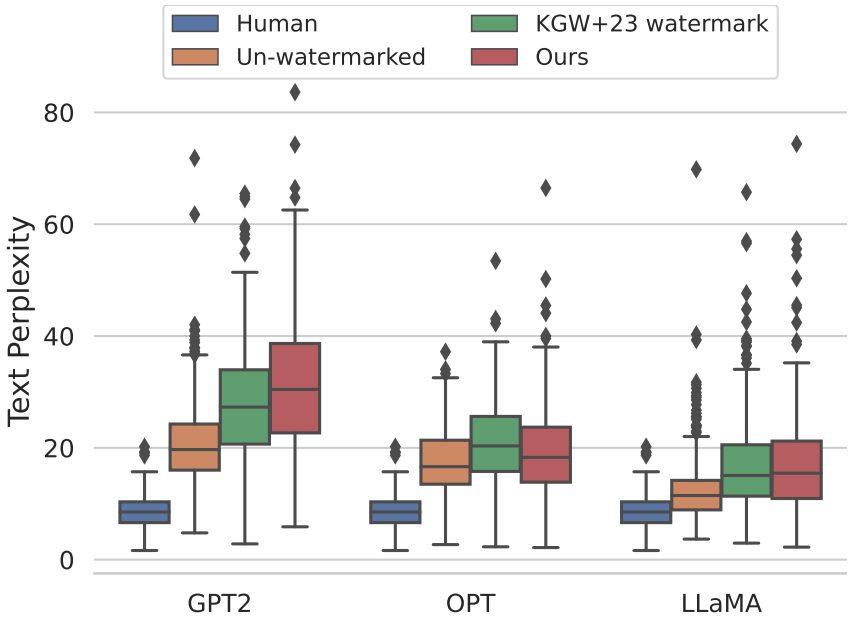

Figure 2.2: Text perplexity comparison (evaluated by GPT-3) between human-generated text and text generated by various models on the OpenGen dataset.

accordingly. We report true positive rate (TPR), F1 score, and ROC curves. GPT3 (`text-davinci-003`) with 175 billion parameters and reinforcement learning from human feedback [51], is used as the oracle model for perplexity evaluation. The experiments are conducted on Nvidia A100 GPUs.

2.4.2 Watermarking results

We use a watermark strength of $\delta = 2.0$ and a green list ratio of $\gamma = 0.5$. We also use different watermark keys `k` for different models. Stronger watermarks can be achieved for shorter sequences for a smaller γ and a larger δ. From the two datasets, we generate 500 watermarked sentences and 500 un-watermarked sentences using three different models (GPT2-XL, OPT-1.3B, and LLaMA-7B). We label them as "watermarked" and "un-watermarked" respectively. We also have corresponding human-written text for each prompt, referred to as "human". All sentences are cropped to a length of 200 tokens.

Setting	Method	OpenGen				LFQA			
		1% FPR		10% FPR		1% FPR		10% FPR	
		TPR	F1	TPR	F1	TPR	F1	TPR	F1
No attack	KGW+23	1.000	0.995	1.000	0.952	1.000	0.995	1.000	0.952
	UNIGRAM-WATERMARK	1.000	0.995	1.000	0.952	1.000	0.995	1.000	0.952
ChatGPT	KGW+23	0.565	0.704	0.853	0.747	0.327	0.453	0.673	0.490
	UNIGRAM-WATERMARK	0.866	0.910	0.961	0.818	0.442	0.568	0.865	0.584
DIPPER-1	KGW+23	0.386	0.546	0.738	0.720	0.372	0.534	0.740	0.767
	UNIGRAM-WATERMARK	0.729	0.830	0.922	0.837	0.639	0.770	0.909	0.865
DIPPER-2	KGW+23	0.490	0.646	0.810	0.769	0.432	0.595	0.845	0.839
	UNIGRAM-WATERMARK	0.777	0.862	0.941	0.852	0.693	0.810	0.948	0.894
BART	KGW+23	0.342	0.505	0.667	0.759	0.457	0.617	0.783	0.836
	UNIGRAM-WATERMARK	0.590	0.730	0.861	0.857	0.656	0.784	0.885	0.897

Table 2.2: Performance comparison of our method (UNIGRAM-WATERMARK) and the soft watermarking method proposed in [26] (denoted as KGW+23). Both methods employ LLaMA-7B with nucleus sampling, utilizing $\delta = 2.0$ and $\gamma = 0.5$. We use ChatGPT, DIPPER, and BART for paraphrasing the watermarked text as paraphrasing attacks. True positive rate and F1 score are presented for fixing the false positive rates at 1% and 10%. When there is no attack, both methods exhibit perfect watermark detection. Nevertheless, when subjected to paraphrasing attacks, UNIGRAM-WATERMARK consistently outperforms KGW+23.

z-scores are calculated for hypothesis testing as shown in Algorithm 2 between different sentence groups. The results (Figure 2.2a) indicate a clear distinction between watermarked and non-watermarked text. A default threshold of z-score $= 6.0$ can be used to determine if a text is watermarked. For a fair comparison with [26], we also set $\delta = 2.0$ and $\gamma = 0.5$ for their method.

Figure 2.2b demonstrates the text perplexity of human, un-watermarked machine-generated, and two watermarking-generated texts, evaluated on the OpenGen dataset. The perplexity of human text is significantly lower, likely due to the expertise contributed in the Wikipedia-based dataset used to train GPT3. We observe that the perplexity of the watermarked text is comparable to that of human-generated text, especially with the use of the largest model LLaMA-7B. This finding further supports the effectiveness

	Avg Score	STD
Un-watermarked	3.660	0.655
Watermarked	3.665	0.619

Table 2.3: Human evaluation result.

of our method in preserving linguistic characteristics and coherence, ensuring seamless integration of watermarks without compromising overall text quality. One example of the prompt questions and machine-generated answers can be found in Table 2.1. We also conduct human evaluations to assess text quality. We enlist crowd workers from Amazon Mechanical Turk (AMT) to evaluate the quality of both watermarked and unwatermarked texts. From the LLaMA-7B model on the OpenGen dataset, we select 100 watermarked and 100 unwatermarked texts, anonymize the sentences, and ask workers to rate the quality on a scale of 1 (poor) to 5 (excellent). Each sentence undergoes two evaluations. The average score and standard deviation are computed and presented in Table 2.3.

2.4.3 Robustness results

Paraphrasing attack. To demonstrate the superior robustness of our method, supported by our theorem, we devise experiments to compare its performance against [26]. We employ different paraphrase attack techniques targeting the removal of the watermark. Firstly, we utilized two versions of the DIPPER model [45], we denote them as "DIPPER-1" and "DIPPER-2". DIPPER-2 has greater diversity than DIPPER-1. Additionally, we leverage the ChatGPT API, generating paraphrased text by providing prompts such as "*Rewrite the following paragraph:*". Furthermore, we employ BART [52] (`bart-large-cnn`, a large-sized model fine-tuned on the CNN Daily Mail dataset [53]) for text summarization as another type of paraphrasing attack. The results of our experiments are shown in Figure 2.4.2 and Table 2.2. The results illustrate the substantial

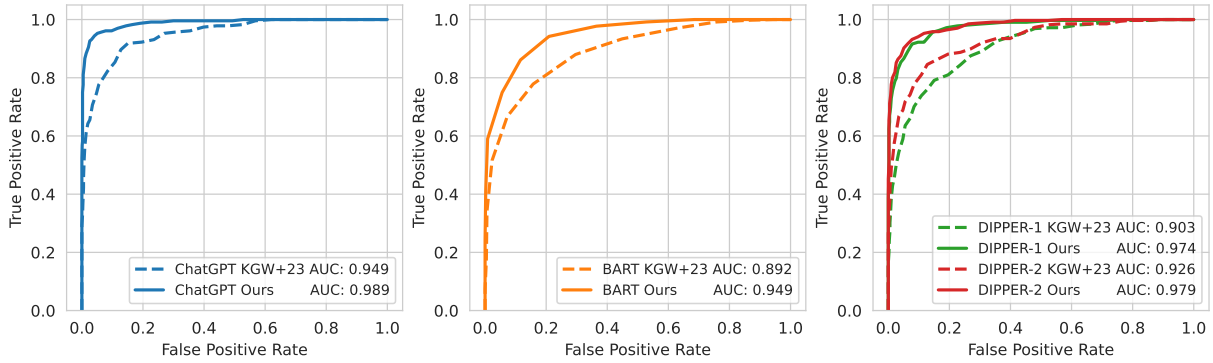

Figure 2.3: UNIGRAM-WATERMARK against paraphrasing attacks on OpenGen dataset with LLaMA-7B.

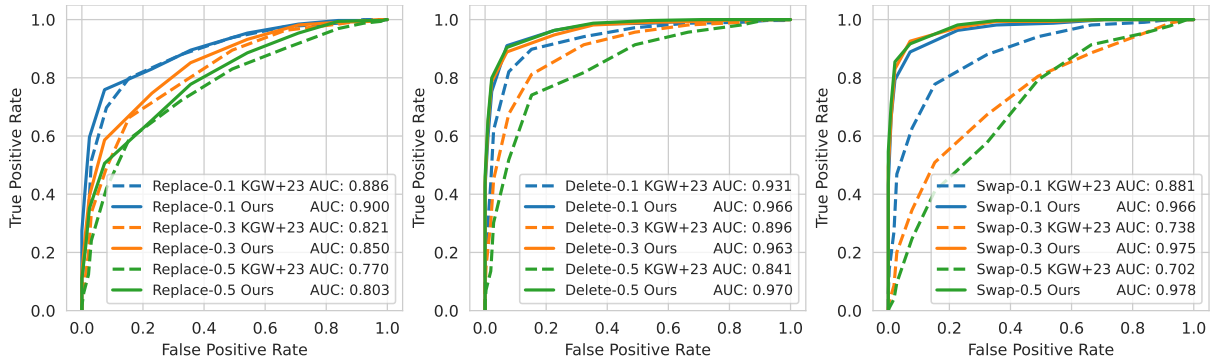

Figure 2.4: UNIGRAM-WATERMARK against editing attacks on LFQA dataset with LLaMA-7B. We vary the rates of synonym replacement, random deletion, and random swapping (0.1, 0.3, 0.5) to demonstrate different attack scenarios.

improvement in robustness achieved by our method compared to [26]. Notably, our method achieves an accuracy rate of over 85% with a false positive rate of 10%.

Editing attack. To further evaluate the robustness of UNIGRAM-WATERMARK against edit attacks, we examine its performance when subjected to synonym replacement, random deletion, and random swapping. These edit attack scenarios represent common techniques used to manipulate text and potentially remove watermarks. We conduct these attacks for the watermarked text of UNIGRAM-WATERMARK and KGW+23. The results are shown in Figure 2.4.2. In each scenario, our method consistently outperforms [26] watermarking scheme, showcasing its enhanced resilience and effectiveness in

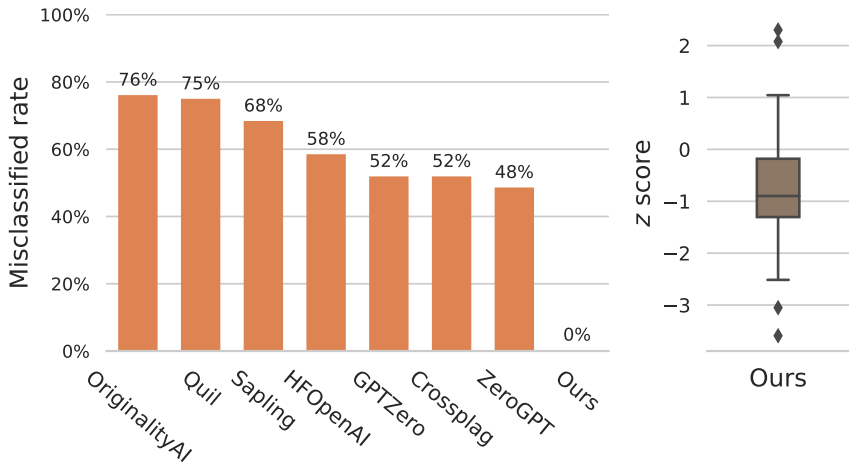

Figure 2.5: Distinguishing human-written text on TOEFL dataset.

protecting the integrity of the embedded watermarks.

2.4.4 Distinguishing human-written text

An interesting observation emphasized by [25] is the misclassification of non-native English writing samples as AI-generated by existing AI content detectors. Our method can effectively establish text origin and maintain robustness to distribution shifts. We evaluate UNIGRAM-WATERMARK in distinguishing human-written text on a dataset of human-written TOEFL essays collected by [25]. Our method demonstrates a remarkable ability to accurately classify human-written text, as evidenced by significantly lower z-scores compared to the empirical threshold of $z = 6.0$. This outcome underscores the effectiveness of our watermark in discerning text generated by human authors, further enhancing its practical utility and reliability.

2.5 Conclusion

In this chapter, we have addressed the concerns surrounding the potential misuse of large language models and proposed an effective watermarking approach, UNIGRAM-

WATERMARK, for detecting machine-generated text from a *specific* language model. Our contributions include the development of a rigorous theoretical framework, designing a provable effective, and robust watermarking scheme under this framework, as well as conducting extensive experiments to demonstrate the effectiveness and robustness of our method in practice. We anticipate that our work will inspire future research to develop more resilient watermarking methods capable of withstanding a broader range of attacks.

Impact statements

Applicability to general K-Gram watermark. While we focused on UNIGRAM-WATERMARK, most of our results apply to K-Gram watermarks with $K \geq 2$ too. These include the Type I error bound, security properties (Robustness to edits), as well as the "Unique" alternative detector which we presented in Appendix 2.6.9. While our Type II error bound does not *directly* work for $K \geq 2$, some of our intermediate steps can be applied.

Limitations. While our watermarking method, UNIGRAM-WATERMARK, demonstrates improved robustness against edits, its reliance on a fixed Green-Red split may not be universally optimal. The performance and robustness of watermarking methods can vary depending on the specific characteristics of the LLM and the generated text. Additionally, although our method enhances detection capabilities, it is not immune to all possible attacks.

Additional experiment results

2.5.1 Empirical error rates

We perform experiments on two datasets (OpenGen and LFQA) using three different models (GPT2-XL, OPT-1.3B, and LLaMA-7B). Table 2.4 presents the error rates, showcasing the sensitivity of the resulting hypothesis test based on observed z-scores. The results demonstrate that there are no Type-I (false positive) errors for all models, with true positive rates exceeding 0.94 for a threshold of $z = 6.0$.

		$z = 6.0$				$z = 7.0$			
Dataset	Model	FPR	TNR	TPR	FNR	FPR	TNR	TPR	FNR
OpenGen	GPT2-XL	0.0	1.0	0.943	0.057	0.0	1.0	0.832	0.168
	OPT-1.3B	0.0	1.0	0.998	0.002	0.0	1.0	0.996	0.004
	LLaMA-7B	0.0	1.0	0.974	0.026	0.0	1.0	0.911	0.089
LFQA	GPT2-XL	0.0	1.0	0.948	0.052	0.0	1.0	0.889	0.111
	OPT-1.3B	0.0	1.0	1.000	0.000	0.0	1.0	0.997	0.003
	LLaMA-7B	0.0	1.0	0.976	0.024	0.0	1.0	0.942	0.058

Table 2.4: Empirical error rates for watermark detection using different models on two datasets. All models employ nucleus sampling with $\delta = 2.0$ and $\gamma = 0.5$. No Type-I (false positive) errors are observed across all models.

2.5.2 Different watermark parameters

We conduct an analysis to understand the impact of changing watermark strength (δ), green list size (γ), and sampling methods on two datasets. The results are summarized in Table 2.5. When using nucleus sampling with a fixed $\gamma = 0.5$, increasing the watermark strength resulted in higher true positive rates (TPR), but it also led to an increase in perplexity (lower quality). Furthermore, for the same watermark strength δ, varying the green list ratio from 0.25 to 0.5 and 0.75 showed improved detection results with smaller γ. Additionally, we explore different decoding methods, transitioning from nucleus sampling

to multinomial sampling and beam search. Remarkably, watermark detection performed effectively with all decoding methods. It is worth noting that the perplexity score for beam search is significantly lower than that of nucleus sampling. However, beam search tends to generate shorter sequences with repeated words.

					$z = 6.0$				$z = 7.0$			
Dataset	decoding	δ	γ	PPL	FPR	TNR	TPR	FNR	FPR	TNR	TPR	FNR
OpenGen	nucleus	1.0	0.5	$18.37_{6.45}$	0.0	1.0	0.576	0.424	0.0	1.0	0.310	0.690
	nucleus	2.0	0.5	$19.42_{8.78}$	0.0	1.0	0.998	0.002	0.0	1.0	0.996	0.004
	nucleus	5.0	0.5	$19.44_{15.02}$	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
	nucleus	10.0	0.5	$19.20_{18.01}$	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
	nucleus	2.0	0.25	$17.96_{9.54}$	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
	nucleus	2.0	0.75	$20.03_{7.67}$	0.0	1.0	0.820	0.180	0.0	1.0	0.485	0.515
	m-nom.	2.0	0.5	$1.75_{0.59}$	0.0	1.0	0.951	0.049	0.0	1.0	0.924	0.076
	4-beams	2.0	0.5	$1.83_{0.97}$	0.0	1.0	0.992	0.008	0.0	1.0	0.982	0.018
	6-beams	2.0	0.5	$1.89_{1.10}$	0.0	1.0	0.984	0.016	0.0	1.0	0.982	0.018
	8-beams	2.0	0.5	$1.96_{1.23}$	0.0	1.0	0.986	0.014	0.0	1.0	0.984	0.016
LFQA	nucleus	1.0	0.5	$18.63_{7.19}$	0.0	1.0	0.455	0.545	0.0	1.0	0.199	0.801
	nucleus	2.0	0.5	$19.14_{11.11}$	0.0	1.0	1.000	0.000	0.0	1.0	0.997	0.003
	nucleus	5.0	0.5	$16.37_{15.39}$	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
	nucleus	10.0	0.5	$16.07_{14.25}$	0.0	1.0	0.998	0.002	0.0	1.0	0.998	0.002
	nucleus	2.0	0.25	$15.27_{10.00}$	0.0	1.0	1.000	0.000	0.0	1.0	1.000	0.000
	nucleus	2.0	0.75	$19.44_{8.20}$	0.0	1.0	0.893	0.107	0.0	1.0	0.582	0.418
	m-nom.	2.0	0.5	$3.17_{2.39}$	0.0	1.0	0.934	0.066	0.0	1.0	0.914	0.086
	4-beams	2.0	0.5	$3.24_{2.85}$	0.0	1.0	0.990	0.010	0.0	1.0	0.986	0.014
	6-beams	2.0	0.5	$3.20_{2.52}$	0.0	1.0	0.994	0.006	0.0	1.0	0.994	0.006
	8-beams	2.0	0.5	$3.13_{2.37}$	0.0	1.0	0.994	0.006	0.0	1.0	0.992	0.008

Table 2.5: Comparison of empirical error rates for watermark detection using nucleus sampling, multinomial decoding, and beam search. Each row represents the average of 500 sequences. While sequences generated with beam search exhibit lower perplexity, they tend to favor shorter outputs, potentially resulting in less diverse text.

2.5.3 Additional robustness results

In addition to the previously discussed robustness evaluations, we provide further analysis of our method's resilience against paraphrasing attacks and editing attacks. The results are presented in Figure 2.7. Notably, our proposed method (UNIGRAM-WATERMARK) consistently outperforms the baseline approach (KGW+23) across various

datasets and attack scenarios. This demonstrates the superior robustness of our method in accurately detecting watermarked text.

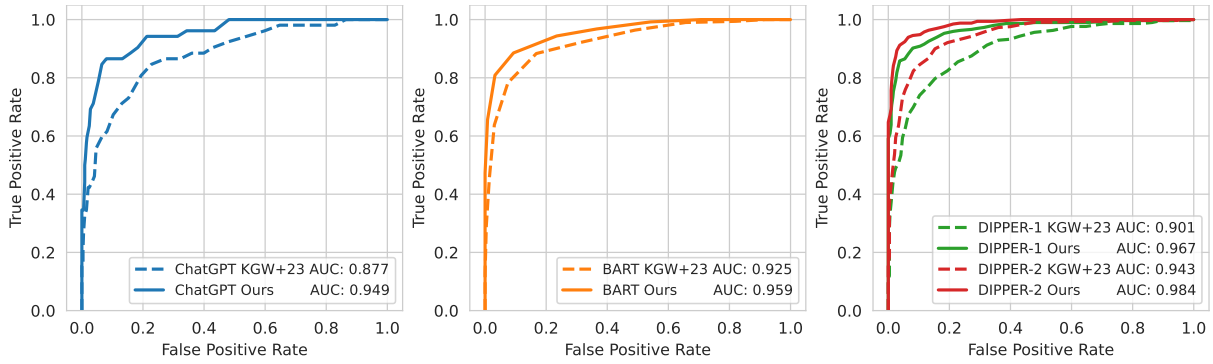

Figure 2.6: UNIGRAM-WATERMARK against paraphrasing attacks on LFQA dataset with LLaMA-7B.

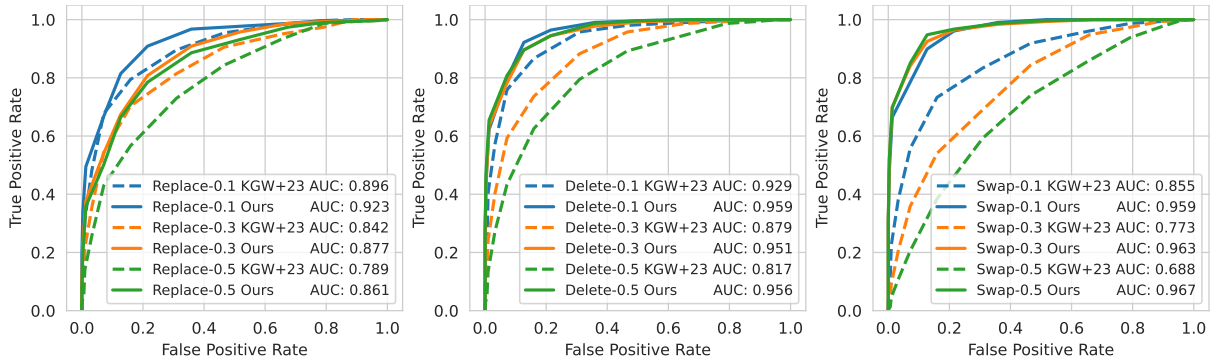

Figure 2.7: UNIGRAM-WATERMARK against editing attacks on OpenGen dataset with LLaMA-7B. We vary the rates of synonym replacement, random deletion, and random swapping (0.1, 0.3, 0.5) to demonstrate different attack scenarios.

2.5.4 White-box attack

A potential attack for UNIGRAM-WATERMARK is to estimate the fixed green and red list. Then the adversary may attempt to bypass detection using these estimated lists. We conduct experiments on white-box attacks and we find that it is difficult to accurately estimate the green list. Even if the green list is known, our watermark is still somewhat

effective thanks to our added robustness.

Estimating the Green List tokens

The question arises: how can the adversary estimate the green list? We simulate an adversary attempting to learn the green list tokens by querying the model multiple times. The adversary collects token distributions from watermarked text and compares them to natural human distributions.

In our experiment, we query the LLaMA-13B watermarked model with watermark strength $\delta = 2.0$, watermark ratio $\gamma = 0.5$ 2500 times, collecting 0.7 million tokens of watermarked text generated from the prompts in LFQA and OpenGen dataset.

Then we simulate three human data distributions:

1. The human response from the same prompt (LFQA and OpenGen dataset). The corresponding human output is 0.4 million tokens. We denote it as the "LFQA & OpenGen dataset"

2. Most times, human responses are not known. So we collect 2000 samples from the C4 [54] dataset to form an approximate human dataset with 1 million tokens. We denote it as the "C4 dataset".

3. To simulate the distribution from non-native speakers. We also collect a non-native speaker (TOEFL essay) dataset from [25] with 12k tokens. We denote it as the "Non-native dataset".

We calculate token frequencies for the three "human" datasets and the watermarked dataset. We use the following decision rule (Algorithm 3) to decide whether a token is green or red.

The estimation results for the green list tokens are shown in the table below.

Algorithm 3 Estimating the Green List tokens

for every token v in the vocabulary $\mathcal{V}$ **do**
  $\Delta(v) \leftarrow$ Frequency(v in watermarked text) $-$ Frequency(v in human text)
  if $\Delta(v) \geq 0$ **then**
    v is in the Green List.
  else
    v is in the Red List.
  end if
end for

Dataset	TPR	FPR	FNR	F1
LFQA & OpenGen dataset	0.692	0.830	0.170	0.755
C4 dataset	0.591	0.806	0.194	0.609
Non-native dataset	0.323	0.923	0.077	0.463

The results suggest that while it is possible to make non-trivial inferences about which token is green, it is hard to say for sure. Notice that we are using a rather big watermark strength. For smaller and more esoteric contexts (prompt, e.g., Non-native TOEFL dataset), such determination is harder.

Evasion attack (white-box and estimated)

In situations where the adversary has either an estimated version or full knowledge of the green and red lists, they can formulate an evasion strategy. We simulate this by assuming the adversary employs WordNet from NLTK to identify token synonyms. Tokens identified as in the green list are replaced with red list synonyms, noting that some tokens may not have synonyms or may only have green synonyms.

The results in Table 2.6 show it is difficult to evade detection even with known green list tokens. The detection AUC for the watermarked text is still somewhat high. In addition, the honest attempt to evade the attack by automatic synonym replacement has led to a significant drop in the text quality.

Green List	Detect AUC	Avg PPL (eval by GPT-3)
No attack	1.000	45.413
Know all green tokens	0.8413	193.410
Estimated from LFQA & OpenGen dataset	0.9397	189.423
Estimated from C4 dataset	0.9291	189.070
Estimated from Non-native dataset	0.9998	125.380

Table 2.6: Evasion attack results: analysis of detection AUC and perplexity.

2.5.5 Testing on scaled language models

	OpenGen	**LFQA**
LLaMA-13B		
No attack	1.000	1.000
ChatGPT attack	0.783	0.854
LLaMA-65B		
No attack	1.000	1.000
ChatGPT attack	0.831	0.697

Table 2.7: Detection results (TPR at 1% FPR) for scaled models LLaMA-13B and LLaMA-65B.

We conduct supplementary experiments on the scaled models LLaMA-13B and LLaMA-65B. Using the same experimental settings as mentioned before, our preliminary results show that our method maintains effectiveness on these larger models. For LLaMA-13B, we are able to use the same test set size. For LLaMA-65B, due to computational constraints, we test on a sample of 100 sentences. The results (TPR at 1% FPR) are shown in Table 2.7.

2.5.6 Results for deduplicated detection

An alternative detector, named "Unique" demonstrates improved robustness in detection and offers advantages in controlling false positives with ease (Section 2.6.9). We conduct experiments to evaluate deduplicated detection performance, with the outcomes

	OpenGen	LFQA
LLaMA-13B		
No attack - Unique Detector	1.000	1.000
ChatGPT attack - Unique Detector	0.679	0.773
LLaMA-65B		
No attack - Unique Detector	1.000	1.000
ChatGPT attack - Unique Detector	0.783	0.682

Table 2.8: Detection results (TPR at 1% FPR) with "Unique" detector.

presented in Table 2.8.

2.6 Proofs of Technical Results

In this section, we state and prove the guarantees for UNIGRAM-WATERMARK which certifies the required quality, correctness, and security properties of a language model watermarking scheme from Definition 2.3.2.

Symbols and mathematical notations. We use $\mathbb{P}[\cdot]$, $\mathbb{E}[\cdot]$, $\mathbb{P}[\cdot|\cdot]$ and $\mathbb{E}[\cdot|\cdot]$ to denote the probability, expectation operator, conditional probability and conditional expectation respectively. Whenever there is ambiguity on which distribution the random variables are drawn from, we explicitly state them, e.g., $\mathbb{P}_{(X,Y)\sim\mathcal{D}}[X < 3|Y = y]$, or equivalently $\mathbb{P}[X < 3|Y = y \ ; \ (X, Y) \sim \mathcal{D}]$. To avoid clutter, we do not distinguish between random variables and constants as the distinctions are clear from the context. Boldface symbols denote a vector, e.g., a probability mass function p or a sequence of tokens y. $\|\cdot\|_2, \|\cdot\|_\infty$ denotes the standard ℓ_2 and ℓ_∞-norms of a vector. In addition, $[n]$ is a shorthand for $\{1, 2, ..., n\}$. Other symbols and their meanings will be defined as we encounter them.

2.6.1 Quality guarantees

We start by providing a strong utility analysis of the watermarked language model than the "perplexity" bound from [26]. Our results work for the entire family of Rényi-divergence and imply guarantees in Kullback-Leibler (KL) divergence and Total Variation-distance.

The Renyi-divergence of two distributions P, Q is defined as

$$D_\alpha\big(P\|Q\big) = \frac{1}{\alpha-1}\log \mathop{\mathbb{E}}_{x\sim Q}\left[(\frac{dP}{dQ})^\alpha\right]$$

where $\frac{dP}{dQ}$ is the Radon–Nikodym derivative. When $\alpha \to 1$, the Renyi divergence converges to the KL-divergence. Additionally, when $\alpha = 0.5$, it serves as an upper bound for the TV-distance.

On the technical level, we leverage a surprising connection to a modern machinery developed in the differential privacy literature known as "bounded range" analysis [44] of the classical exponential mechanism [55].

Theorem 2.6.1 (Restatement of Theorem 2.3.4)**.** *Consider h as the input to the language model at step t, denoted as $h = [x, y_{1:t-1}]$. Fix green list G. Let δ represent the watermark strength. For any h, the α-th order Renyi-divergence between the watermarked probability distribution $\hat{p}_t = \hat{p}_t(\cdot|h)$ at time step t and the original probability distribution $p_t = p_t(\cdot|h)$ satisfies:*

$$\forall h, \max\left(D_\alpha\big(\hat{p}_t\|p_t\big), D_\alpha\big(p_t\|\hat{p}_t\big)\right) \le \min\{\delta, \alpha\delta^2/8\}.$$

Proof. We define $\delta_v = 0$ when $v \in R$ and $\delta_v = \delta$ when $v \in G$. Using this definition, we have:

$$\hat{p}(v|h) = \frac{\exp(\boldsymbol{\ell}_v + \delta_v)}{\sum_w \exp(\boldsymbol{\ell}_w + \delta_w)} \le \frac{\exp(\delta)\exp(\boldsymbol{\ell}_v)}{\exp(-\delta)\sum_w \exp(\boldsymbol{\ell}_w)} = e^{2\delta}p(v|h)$$

Similarly, $\hat{p}(v|h) \geq e^{-2\delta} p(v|h)$.

Consequently, $\hat{p}$ and p are 2δ-close in terms of max-divergence, which can be interpreted as $(\epsilon, \tilde{\delta})$-indistinguishable, similar to the concept of Differential Privacy [43] with $\tilde{\delta} = 0$ and $\epsilon = 2\delta$.

Additionally, $\hat{p}(v|h)$ and $p(v|h)$ satisfy δ-BoundedRange (Proposition 1 in [44]) with parameter δ, since the changes to $\boldsymbol{\ell}_v$ is monotonic. Lemma 3.2 in [56] shows that δ-Bounded Range implies $\delta^2/8$-concentrated differential privacy, which says that $D_\alpha(\hat{p}\|p) \leq \frac{\delta^2 \alpha}{8}$ for all $\alpha \geq 1$ (where D_α represents Rényi Divergence of order α). Specifically, when $\alpha = 1$, the KL-divergence satisfies $D_{\mathrm{KL}}(\hat{p}\|p) \leq \frac{\delta^2}{8}$.

Furthermore, δ-BoundedRange implies δ-DP (or rather $(\delta, 0)$-indistinguishability, since we are dealing with just two distributions rather than a family of neighbor distributions). It follows from the that

$$D_{\mathrm{KL}}(\hat{p}\|p) \leq D_\infty(\hat{p}\|p) \leq \delta$$

□

Corollary 2.6.2. *For any prompt x, the KL-divergence between the probability distribution of the watermarked sequence and the original sequence satisfies:*

$$\forall x, \max\{D_{\mathrm{KL}}\big(\hat{p}(y_{1:n}|x)\|p(y_{1:n}|x)\big), D_{\mathrm{KL}}\big(p(y_{1:n}|x)\|\hat{p}(y_{1:n}|x)\big)\} \leq \alpha \min\{n\delta, n\delta^2/8\}$$

Proof. The proof follows from the adaptive composition theorem for Renyi-divergence, and max-divergence (from the DP literature) for the autoregressive decomposition of $\hat{p}(y_{1:n}|x)$ and $p(y_{1:n}|x)$ and then invoke Theorem 2.3.4 for each factor. □

2.6.2 Robustness / Security guarantees

In this section, we provide the proof for Theorems 2.3.10, 2.6.24, and 2.3.4 to ensure completeness and precision. We begin by restating the theorems and providing the corresponding proofs with necessary modifications.

Theorem 2.6.3 (Robustness to editing (Restatement of Theorem 2.3.10))**.** *Let $y = [y_1, \ldots, y_n]$ represent the watermarked sequence. Suppose the adversary $\mathcal{A}$ follows Definition 2.3.2 and outputs a modified text $u = [u_1, \ldots, u_m]$. Following Equation 2.2, we calculate z-score z_y and z_u. Assume edit distance between y and u (denoted as η) satisfies $\eta < n$. Then we have*

$$z_u \geq z_y - \max\{\frac{(1+\gamma/2)\eta}{\sqrt{n}}, \frac{(1-\gamma/2)\eta}{\sqrt{n-\eta}}\}.$$

In particular, when $\eta \leq \frac{2\gamma n}{(1+\gamma/2)^2}$, we can drop the second term in the max.

Proof. Define bivariate function $f(x, y) = \frac{x-\gamma y}{\sqrt{y}}$. By Taylor's theorem

$$f(x-k_x, y-k_y) = f(x,y) + \begin{bmatrix} \partial_x f(x-\tilde{k}_x y - \tilde{k}_y) \\ \partial_y f(x-\tilde{k}_x y - \tilde{k}_y) \end{bmatrix}^T \begin{bmatrix} -k_x \\ -k_y \end{bmatrix} = f(x,y) - \left(\frac{k_x}{\sqrt{y-\tilde{k}_y}} - \frac{\gamma k_y}{2\sqrt{y-\tilde{k}_y}} \right)$$

where $\tilde{k}_x$ is between 0 and k_x and $\tilde{k}_y$ is between 0 and k_y. We also know that $|k_x| \leq k$ and $|k_y| \leq k$.

A lower bound of the above can be obtained by finding an upper bound to

$$\frac{k_x}{\sqrt{y-\tilde{k}_y}} - \frac{\gamma k_y}{2\sqrt{y-\tilde{k}_y}} = \frac{k_x - \frac{\gamma}{2}k_y}{\sqrt{y-\tilde{k}_y}}$$

First observe that we can always choose $k_x = k$. Next we discuss two possibilities of k_y. If k_y is negative, then choosing $k_y = -k$ and $\tilde{k} = 0$ maximizes the bound, which gives

$\frac{(1+\gamma/2)k}{\sqrt{y}}$.

If k_y is positive, then we should always choose $\tilde{k}_y = k_y$ to maximize the expression, which gives us an upper bound of

$$\frac{k - \frac{\gamma}{2}k_y}{\sqrt{y - k_y}} = \frac{k + \frac{\gamma}{2}(y - k_y) - \frac{\gamma}{2}y}{\sqrt{y - k_y}} = \frac{k - \frac{\gamma}{2}y}{\sqrt{y - k_y}} + \frac{\gamma\sqrt{y - k_y}}{2}.$$

We will discuss two cases again, the first case is when $k - \gamma y/2 \leq 0$. In this case, the function $g(u) = a/u + bu$ with $a \leq 0$ has a derivative of $-a/u^2 + b \geq 0$, thus g is monotonically increasing. Thus we should choose $k_y = 0$. The second case is when $k - \gamma y/2 > 0$, in this case the $a > 0$ in the above $g(u)$ and $g(u)$ is convex, thus $\max_{u_{\min} \leq u \leq u_{\max}} g(u) = \max\{g(u_{\max}), g(u_{\min})\}$. Thus we should just compare the two cases when $k_y = 0$ and $k_y = k$, i.e., $\max\{\frac{k}{\sqrt{y}}, \frac{(1-\gamma/2)k}{\sqrt{y-k}}\}$.

Collect everything together, we get an upper bound o

$$\max\{\frac{(1+\gamma/2)k}{\sqrt{y}}, \frac{k}{\sqrt{y}}, \frac{(1-\gamma/2)k}{\sqrt{y-k}}\} = \max\left\{\frac{(1+\gamma/2)k}{\sqrt{y}}, \frac{(1-\gamma/2)k}{\sqrt{y-k}}\right\}$$

i.e.,

$$f(x - k_x, y - k_y) - f(x, y) \geq -\max\left\{\frac{(1+\gamma/2)k}{\sqrt{y}}, \frac{(1-\gamma/2)k}{\sqrt{y-k}}\right\}.$$

Now notice that our z-score has the same form as the $f(x, y)$ function. We can take $y = n$ and $x = |y|_G$. Instantiate k be the maximum number of edits η. Observe that given that the adversary has a bounded edit distance, each operation of "insertion", "deletion", or "edit" can, at most, alter one token from the green list to the red list. They also can only alter the length by the number of edits. The above result translates into

$$z_u \geq z_y - \max\{\frac{(1+\gamma/2)\eta}{\sqrt{n}}, \frac{(1-\gamma/2)\eta}{\sqrt{n-\eta}}\},$$

where η denotes the edit distance between y and u. □

The robustness theorem above implies the security guarantees as we discussed in Corollary 2.6.23.

2.6.3 No false positive (Type I error guarantees)

Theorem 2.6.4 (No false positives)**.** *Consider $y = y_{1:n}$ as any* fixed *suspect text. Let $N =: |\mathcal{V}|$ and $G \subset |\mathcal{V}|$ satisfying $|G| = \gamma N$. G is selected through Algorithm 1, using a uniform random choice. Let $|y|_G$ denote the number of tokens in G and $z_y := \frac{|y|_G - \gamma n}{\sqrt{n\gamma(1-\gamma)}}$ as in Algorithm 2. Then the following statements hold true:*

1. *Assume $n \geq 1$, then*

$$\mathbb{E}[|y|_G|y] = \gamma n \quad \text{and} \quad \mathbb{E}[z_y|y] = 0.$$

2. *Define $C_{\max}(y) := \max_{i\in[N]} \sum_{j=1}^{n} \mathbf{1}(y_j = i)$ and $V(y) := \frac{1}{n}\sum_{i=1}^{N}(\sum_{j=1}^{n} \mathbf{1}(y_j = i))^2$, then with probability $1-\alpha$ (over only the randomness of G),*

$$\mathbb{P}\left[|y|_G \geq \gamma n + \sqrt{64\gamma n V \log(9/\alpha)} + 16 C_{\max}\log(9/\alpha)\middle| y\right] \leq \alpha$$

or equivalently (when $n \geq 1$)

$$\mathbb{P}\left[z_y \geq \sqrt{\frac{64V\log(9/\alpha)}{1-\gamma}} + \frac{16 C_{\max}\log(9/\alpha)}{\sqrt{n\gamma(1-\gamma)}}\middle| y\right] \leq \alpha.$$

Proof. To prove the first statement, observe that any fixed token has a probability γ to be included in the green list, thus by the linearity of the expectation and the independence

of y in G.

$$\mathbb{E}[|y|_G|y] = \sum_{i=1}^{n} \mathbb{E}[\mathbf{1}(y_i \in G)|y] = \sum_{i=1}^{n} \gamma = \gamma n.$$

Next, we will prove the second statement by applying Lemma 2.6.28 to obtain the result stated in the third statement. Let $a_{i,j} = \mathbf{1}(j \leq \gamma N) \sum_{\ell=1}^{n} \mathbf{1}(y_\ell = i)$. By our assumption $0 \leq a_{i,j} \leq C_{\max}$ for all i, j. Observe that $\sum_{i=1}^{N} a_{i,\Pi_N(i)}$ is *identically distributed* with $|y|_G$.

By Lemma 2.6.28 with $t = 16 \log(8e^{1/16}/\alpha)$, we get that with probability $1 - \alpha$,

$$||y|_G - \gamma n| < 2\sqrt{\frac{16 \log(9/\alpha)}{N} N\gamma n V} + 16 C_{\max} \log(9/\alpha)$$

where we used that $8e^{1/16} \leq 9$ and the fact that only γN columns of the $a_{i,j}$ matrix $a_{i,j}$ is nonzero, and for each non-zero column L2-norm of the column is bounded by $\sqrt{nV}$ by our definition of V. The result for the z-score follows trivially. □

Remark 2.6.5 (Wide applicability). Note that the theorem does not impose assumptions on how y is generated. It covers any procedure (including human generation) that produces y in a manner *independently* of the secret partition G. In cases where y is generated by a language model, it could be the output of greedy search from $p(y_t|x, y_{1:t-1})$, nucleus sampling, beam search, or any other decoding methods.

Remark 2.6.6 (Diversity parameters). The V and $C_{\max}$ parameters in Theorem 2.6.4 measure the *diversity* of the suspect text y and are necessary for the high-probability bound. As an example, if the prompt says "`Repeat ''Goal'' for a hundred thousand times like a soccer commentator.`" Then the resulting generated sequence will be "`Goal goal goal ...`", and has either n green tokens or 0 green tokens. No meaningful Type I error bound can be obtained.

Remark 2.6.7 (Controlling false positive rate). The theorem implies that if we choose

$\tau > \sqrt{\frac{64V\log(9/\alpha)}{1-\gamma}} + \frac{16C_{\max}\log(9/\alpha)}{\sqrt{n\gamma(1-\gamma)}}$, then the false-positive rate is smaller than α. Note that V and $C_{\max}$ can be computed directly from y, allowing us to choose an input-dependent τ as a function of $V, C_{\max}$ that achieves a α-Type I error guarantee with a fixed α for all inputs. In particular, the Type I error α decreases exponentially as we increase the threshold τ.

2.6.4 Only true detection (Type II error guarantees)

For bounding the Type II error, i.e., false negative rates, we will work with our proposed method that generates y from the language model, i.e., sampling from the watermarked distribution $\hat{p}$ recursively one token at a time.

Let's first recall a few notations. h is the input to the language model at step t, i.e., $h = [x, y_{1:t-1}]$. Let δ represent the watermark strength from Equation 2.1. The green list $G \subset [N]$ is a random index set of the vocabulary of size γN. The watermarked probability distribution $\hat{p}_t = \hat{p}_t(\cdot|h)$ at time step t. The process of generating the sentence $y_1, y_2, \ldots, y_n$ involves recursively sampling from $\hat{p}_t$, which we refer to as a "roll-out" procedure.

We need to make a few assumptions about the language model's probability distribution p and the prompt x. We will first state them and then explain why these are natural and arguably needed for the Type II error to be small.

On-average high entropy assumption

The first such assumption requires the probability of the roll-out to be "sufficiently diverse" on average. We will introduce the notation $\|p\|_2 := \sqrt{\sum_{i=1}^{N} p[i]^2}$.

Assumption 2.6.8 (On-average-high-entropy)**.** We say a language model's probability

distribution p with a prompt x satisfies ξ-on-average-high-entropy if

$$\frac{1}{n}\sum_{t=1}^{n} \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)}[\|p_t\|^2] \leq \xi.$$

This assumption requires the distribution of the roll-out to be sufficiently diffuse on average (either in expectation or with high probability).

The purpose of these assumptions is to rule out the cases when $y_{1:n}$ is almost deterministic under p and perturbing the logits by δ does not change the distribution much at all.

For example, if the prompt writes

"`Generate the English alphabet in capital letters for 200 times please.`"

Then the language model would generate

"`ABC...XYZ, ABC...XYZ, ...`".

Despite that the generated sequence is very long, i.e., n is as large as $5,200$, the added watermark does not change the distribution very much at all. To see this, if $p(y_3 =$ "C"$|x, h) \geq 1 - \epsilon$ for a tiny ϵ, and then by our quality guarantee, $\hat{p}(y_3 =$ "C" $|x, h) \geq 1 - \epsilon e^{\delta}$.

Quantitatively, for nearly uniform p_t, $\xi = O(1/N)$, if p_t concentrates on a single token for all t, e.g., when a football commentator exclaims "`Goal goal goal goal ....`", then we cannot obtain a better bound than the trivial $\xi \leq 1$. In the alphabet example above $\xi \leq 1/26$.

Why is it called entropy? Assumption 2.6.8 is related to the "high-entropy" assumption in [26] but for a slightly different kind of entropy. In a more formal sense, the quantity $\|p_t\|^2$ is connected to the Tsallis entropy of order 2, defined as $S_2(p_t) = k_B(1 - \|p_t\|^2)$ where k_B is known as the Boltzmann constant. Our assumption requires the expected

Tsallis entropy of the conditional distribution p_t over the roll-out of p to be larger than $k_B(1-\xi)$ on average among $t = 1, ..., n$.

For a high-probability result, we also need a stronger version.

Assumption 2.6.9 (On-average-high-entropy (high probability))**.** We say that a language model's probability distribution p with a prompt x satisfies (ξ, β)-on-average-high-entropy if with probability at least $1-\beta$ over the generated sequence $y_{1:n}$,

$$\frac{1}{n}\max\left\{\left\|\sum_{t=1}^{n} p_t\right\|, \sum_{t=1}^{n}\|p_t\|^2, \left\|\sum_{t=1}^{n} p_t\right\|_\infty, \sum_{t=1}^{n}\|p_t\|_\infty^2\right\} \leq \xi.$$

The behavior is similar to that of the expectation version of the assumption. When p_t is nearly uniform, $p_t[i] = O(1/N)$, then $\xi = O(1/\sqrt{N})$. When p_t is supported only on one token, then $\xi = 1$. In practice, ξ is a small constant. As we will present in the main theorem, as long as $\xi \asymp \delta$, the number of green list tokens is guaranteed to grow faster γn as n gets larger.

One may also ask whether it is necessary to make entropy assumptions on the conditional probabilities instead of the marginal probabilities induced by p or $\hat{p}$, but this is unfortunately not sufficient as illustrated in the following example.

Example 2.6.10 (Marginal high entropy is insufficient)**.** Let the prompt x be

"`Generate the first token uniformly at random, then repeat the token you generated for the remaining` $n-1$ `tokens`".

In this case, a good language model that follows the instruction will have $\mathbb{P}_p(y_t = i) = 1/N$ for all i and all $t = 1, ..., n$ marginally, which implies that the entropy is the maximum and for any green list G, $\mathbb{P}_p(y_t \in G) = \gamma$. On the other hand, with probability γ, $|y|_G = n$ and with probability $1-\gamma$, $|y|_G = 0$. There isn't any concentration around γn possible. Moreover, check that if we apply watermark, then $\mathbb{P}_{\hat{p}}(y_t \in G) = \frac{\gamma e^\delta}{\gamma e^\delta + (1-\gamma)}$ for all t and

all G. This changes the probability of seeing $|y|_G = n$ slightly but the two world remains indistinguishable.

A "homophily" assumption

The second assumption that we need to make is called "homophily", which says that increasing the probability of a group of tokens by adding the watermarks will not decrease the probability of generating the same group of tokens in the future as the language model rolls out.

Assumption 2.6.11 ("Homophily")**.** We say a language model's probability distribution p and prompt x satisfy "homophily" if for any G, the corresponding watermarked $\hat{p}$ satisfies that

$$\mathop{\mathbb{E}}_{h\sim\hat{p}(\cdot|x)}\left[\mathop{\mathbb{P}}_{y\sim\hat{p}(\cdot|h,x)}(y\in G)\right] \geq \mathop{\mathbb{E}}_{h\sim p(\cdot|x)}\left[\mathop{\mathbb{P}}_{y\sim\hat{p}(\cdot|h,x)}(y\in G)\right]$$

where h denotes the generated sequence before y.

This assumption says that by increasing the probability of tokens in G, the induced distribution of the prefix h cannot counter-intuitively reduce the probability of tokens in G in the future on average.

The assumption is not unreasonable, because we expect a language model to be more likely to refer to text it has generated in the prefix than those that did not appear in the prefix.

This "homophily" assumption is needed to rule out the unnatural situation where increasing the green list tokens initially ends up reducing the number of green list tokens in the long run. To illustrate this, consider the following example utilizing the prompt:

$x =$ "`Randomly select a color, state what it is. Then write a short poem about it without naming this color at all.`"

The generated text from a commercial language model is

"`Color choice: green. Emerald whispers in the meadow's sway, Life's verdant rhythm in ceaseless play. It cradles the world in a leafy embrace, A silent serenade to nature's grace.`"

Notice that if the token "`green`" $\in G$, it increases the probability of the language model generating "`green`" at the beginning. However, regardless of the text's length, the subsequent portion of the generated text will not contain the word "`green`", as instructed by the prompt. This decreases the expected number of times the token "`green`" appears.

To hammer it home, consider the following more quantitative construction of that works no matter which random green list G realizes.

$x =$ "`Choose the first k token by random sampling without replacement. Then sample from all but the token you choose uniformly for n-k rounds.`"

It's easy to calculate that the expected number of times any token appears in a language model that perfectly follows the instruction will be n/N. However, the watermarked language model, let's say we use a very large δ such that the first k tokens are from the green list, then the expected number of times a green-list token appears is $\frac{k}{\gamma N} + \frac{\gamma N - k}{\gamma N} \frac{(n-k)(\gamma N - k)}{N-k}$ which is bounded by 1 if $k = \gamma N$ instead of growing linearly in n as in the original language model.

To obtain a concentration bound, we also need a stronger version of the homophily assumption as follows.

Assumption 2.6.12 (High probability on-average homophily)**.** There exists a coupling – a joint distribution of $y_{1:n}$ and $\hat{y}_{1:n}$ where marginally $y_{1:n} \sim p(\cdot|x)$, $\hat{y}_{1:n} \sim \hat{p}(\cdot|x)$ – such

that for any G, with probability $1-\beta$ over the joint distribution,

$$\frac{1}{n}\sum_{t=1}^{n}\hat{p}_t(G|\hat{y}_{1:t-1})) \geq \frac{1}{n}\sum_{t=1}^{n}\hat{p}_t(G|y_{1:t-1})).$$

The reason for defining the existence of a coupling is for technical reasons, but the purpose of the assumption is identical to that of the in-expectation version.

2.6.5 Theorem statement on "Only true detection"

Now we are ready to state the main theorem.

Theorem 2.6.13 (Only true detection)**.** *For a fixed language model $\mathcal{M}$ and a prompt x. The sentence $y_{1:n}$ generated from $\hat{\mathcal{M}}(x)$ where $\hat{\mathcal{M}}$ is an output of our watermarking scheme* $\mathsf{Watermark}_{\delta,\gamma}(\mathcal{M})$ *with parameter δ, γ. Then the following statements are true.*

1. *Assume homophily (Assumption 2.6.11), then*

$$\mathbb{E}[|y|_G] \geq \frac{n\gamma e^{\delta}}{1+(e^{\delta}-1)\gamma} - \gamma(1-\gamma)e^{\delta}\sum_{t=1}^{n}\mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)}\|p_t\|^2.$$

   In particular, if Assumption 2.6.8 condition is true with parameter $\xi \leq (1-\kappa)\frac{e^{\delta}-1}{(1+(e^{\delta}-1)\gamma)e^{\delta}}$ for a parameter $0<\kappa<1$, then

$$\mathbb{E}[|y|_G] \geq n\gamma\left(1+\kappa\frac{(e^{\delta}-1)(1-\gamma)}{1+(e^{\delta}-1)\gamma}\right) \text{ or equivalently } \mathbb{E}[z_y] \geq \frac{\kappa(e^{\delta}-1)\sqrt{n\gamma(1-\gamma)}}{1+(e^{\delta}-1)\gamma}.$$

2. *Assume high-probability version of homophily (Assumption 2.6.12). There exists a parameter $C_{\delta,\gamma}$ that depends only δ, γ such that with probability at least $1-\beta$ for any*

$\beta > 0$ (over both G and $y \sim \hat{p}(\cdot|x, G)$),

$$\|y\|_G \geq \frac{n\gamma e^\delta}{1+(e^\delta-1)\gamma} - \sqrt{2n\log(6/\beta)} - C_{\delta,\gamma}\log^2\frac{27(n+1)}{\beta}\left(\|\sum_{t=1}^n p_t\| + \sum_{t=1}^n \|p_t\|^2 + \|\sum_{t=1}^n p_t\|_\infty + \sum_{t=1}^n \|p_t\|_\infty^2\right).$$

In particular, if for a parameter $0 < \kappa < 1$,

$$n \geq \frac{8\log(6/\beta)(1-\gamma+e^\delta\gamma)^2}{(1-\kappa)^2\gamma^2(1-\gamma)^2(e^\delta-1)^2} = \tilde{\Omega}(1/\delta^2) \tag{2.3}$$

and Assumption 2.6.9 condition is true with parameter $(\xi, \beta/3)$ where

$$\xi \leq \frac{(1-\kappa)\gamma(1-\gamma)(e^\delta-1)}{8C_{\delta,\gamma}(1-\gamma+e^\delta\gamma)\log^2\left(\frac{27(n+1)}{\beta}\right)} = \tilde{O}(\delta), \tag{2.4}$$

then

$$\mathbb{P}\left[\|y\|_G < n\gamma(1+\kappa\frac{(e^\delta-1)(1-\gamma)}{1-\gamma+\gamma e^\delta})\right] = \mathbb{P}\left[z_y < \frac{\kappa(e^\delta-1)\sqrt{n\gamma(1-\gamma)}}{1+(e^\delta-1)\gamma}\right] \leq \beta.$$

Remark 2.6.14 (Exponentially small Type I and Type II error guarantees). Recall that according to Theorem 2.6.4, in order to have a false positive rate controlled at level α, we need to set the threshold $\tau \gtrsim \sqrt{\log(1/\alpha)}$ for sufficiently high-entropy sequences. Theorem 2.6.13 says that if we want the false negative rate to be smaller than β, we only need the threshold $\tau \lesssim \kappa\delta n$ under similar (slightly different) high-entropy sequences for $n \gtrsim \log(1/\beta)/\delta^2$. Observe that there is a wide range of valid choices of τ for us to have a detection algorithm that does not make Type I or Type II error with high probability. These observations together suggest that we can afford to choose $\delta \asymp 1/\sqrt{n}$ if the sequence is sufficiently high-entropy.

Remark 2.6.15 (Information-theoretic optimality). The sample complexity of $n \gtrsim 1/\delta^2$ is information-theoretically optimal (up to a logarithmic factor) in δ because, our accuracy guarantee (together with the composition theorem) indicates that the KL-divergence between a sequence of length n generated from p and that generated from $\hat{p}$ is $n\delta^2$ indistinguishable, i.e., $n > 1/\delta^2$ for any classifier — even the uniform most-powerful Neyman-Pearson likelihood-ratio test (which requires additional information, e.g., x and p which we do not have) — to make no mistakes with a constant probability.

2.6.6 Proof of Theorem 2.6.13

In the false negative error cases, y is drawn from the watermarked language model $\hat{\mathcal{M}}$. To be explicit, let us write $y = [\hat{y}_1, ..., \hat{y}_n] = \hat{y}_{1:n}$. Now let's also define a hypothetical (possibly coupled) sequence $y_{1:n}$ which is drawn from the original (un-watermarked) language model $\mathcal{M}$.

For convenience, we define the following shorthand $p(G) := \mathbb{P}_{y\sim p}[y \in G]$. for a probability mass function p defined on the vocabulary $\mathcal{V}$. Specifically, $\hat{p}_t(G|\hat{y}_{1:t-1})$ means $\mathbb{P}_{y\sim\hat{p}_t(\cdot|x,\hat{y}_{1:t-1})}[y \in G]$, parameterized by a fixed green list G. Similarly, $p_t(G|y_{1:t-1})$ denotes $\mathbb{P}_{y\sim p_t(\cdot|x,y_{1:t-1})}[y \in G]$.

The proof of Theorem 2.6.13 considers the following decomposition

$$|y|_G = |y|_G - \sum_t \hat{p}_t(G|\hat{y}_{1:t-1}) \tag{2.5}$$

$$+ \sum_t \hat{p}_t(G|\hat{y}_{1:t-1}) - \sum_t \hat{p}_t(G|y_{1:t-1}) \tag{2.6}$$

$$+ \sum_t \hat{p}_t(G|y_{1:t-1}) \tag{2.7}$$

steps to prove a lower bound to each of the three terms. We will start with the high probability bound (the second statement in Theorem 2.6.13) then deal with the expec-

tation.

Many green list tokens with high probability

To obtain a high-probability lower bound, it requires us to obtain concentration for each of the three terms. Specifically,

1. To bound Term (2.5), we use Lemma 2.6.16 which invokes Martingale concentration over the randomness in y to show $|y|_G$ is close to $\sum_t \hat{p}_t(G|\hat{y}_{1:t-1})$.

2. We will show Term (2.6) is non-negative with high probability by using the homophily assumption (Assumption 2.6.12). This allows us to study the roll-out $\hat{y}_{1:t-1}$ under $\hat{\mathcal{M}}(x)$ (or $\hat{p}$) by studying a hypothetical alternative roll-out $y_{1:t-1}$ sampled under $\mathcal{M}(x)$ (or p).

3. Then we control Term (2.7) by first Taylor expanding it into quantities involving $p_t(G|y_{1:t-1})$ instead of $\hat{p}(G|y_{1:t-1})$, then apply concentration inequalities for each expanded terms over the randomness of G (while fixing $y_{1:t-1}$) to obtain a high probability lower bound. Proposition 2.6.19 gives the results.

We start by tackling (2.5) via Martingale concentration.

Lemma 2.6.16. *For any green list G and prompt x.*

$$\mathbb{E}\left[|y|_G - \sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim \hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G]\right] = 0.$$

Moreover, with probability at least $1-\beta$ over the roll-out

$$|y|_G \geq \sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim \hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G] - \sqrt{2n\log(2/\beta)}.$$

Proof. We fix G and construct a martingale sequence $X_1, X_2, ..., X_n$ where $X_0 = 0$ and:

$$X_t = X_{t-1} + \mathbf{1}(y_t \in G) - \mathop{\mathbb{P}}_{y_t \sim \hat{p}(\cdot|x, y_{1:t-1})}[y_t \in G].$$

Check that $\mathbb{E}[X_t|y_{1:t-1}] = X_{t-1}$. The underlying filtration is the sigma-field generated by $y_{1:t}$.

The claim about the expectation follows from that $X_0 = 0$ and an inductive argument following the tower property of conditional probabilities.

By the fact that $|X_t - X_{t-1}| \leq 1$ we can apply Azuma-Hoeffding's inequality and get

$$\mathbb{P}\left[|X_n - \mathbb{E}[X_n]| \geq u\right] \leq 2e^{-\frac{u^2}{2n}}.$$

Check that by an inductive argument $\mathbb{E}[X_n] = 0$. So we get that with probability at least $1 - \delta$

$$|X_n| = \left| \sum_{t=1}^{n} \mathbf{1}(y_t \in G) - \sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim \hat{p}(\cdot|x, y_{1:t-1})}[y_t \in G] \right| \leq \sqrt{2n \log(2/\delta)}.$$

□

To handle (2.6), we apply Assumption 2.6.12 with parameter $\beta/3$, which says that with probability $1 - \beta/3$ (2.6)≥ 0. This converts a roll-out from $\hat{y} \sim \hat{p}$ to a roll-out from the original p.

Before we deal with (2.7), let us write a lemma that rewrites $\hat{p}_t(G|y_{1:t-1})$ into a more convenient form.

Lemma 2.6.17. *For any t, h_t. Fix G. Denote short hands $\hat{p}(G) := \mathbb{P}_{y_t \sim \hat{p}_t(\cdot|x, h_t)}[y_t \in G]$ and $p(G) := \mathbb{P}_{y_t \sim p_t(\cdot|x, h_t)}[y_t \in G]$.*

$$\hat{p}(G) = \frac{e^\delta p(G)}{1 + (e^\delta - 1)p(G)} = \left(1 + \frac{(e^\delta - 1)(1 - p(G))}{1 + (e^\delta - 1)p(G)}\right) p(G).$$

Proof. By definition,

$$\begin{aligned}\hat{p}(G) &= \frac{\sum_{y\in G} e^{\ell_y+\delta}}{\sum_{y\in G} e^{\ell_y+\delta} + \sum_{y\notin G} e^{\ell_y}} \\ &= \frac{e^\delta p(G)}{e^\delta p(G) + 1 - p(G)} = \frac{e^\delta}{1+(e^\delta-1)p(G)} p(G) \\ &= \left(1 + \frac{(e^\delta-1)(1-p(G))}{1+(e^\delta-1)p(G)}\right) p(G).\end{aligned}$$

□

The lemma implies that $\hat{p}(G) \geq p(G)$ and that if $p(G)$ is bounded away from 1, $\hat{p}(G) \geq (1+O(\delta))p(G)$.

Lemma 2.6.18. *For any t, h_t. Fix G.*

$$\hat{p}(G) \geq \frac{e^\delta\gamma}{1+(e^\delta-1)\gamma} + \frac{e^\delta}{(1+(e^\delta-1)\gamma)^2}(p(G)-\gamma) - e^\delta(p(G)-\gamma)^2$$

Proof. By the second-order Taylor's theorem

$$\frac{e^\delta x}{1+(e^\delta-1)x} = \frac{e^\delta\gamma}{1+(e^\delta-1)\gamma} + \frac{e^\delta}{(1+(e^\delta-1)\gamma)^2}(x-\gamma) - \frac{e^\delta}{(1+(e^\delta-1)\tilde{x})^3}(x-\gamma)^2$$

where $\tilde{x} \in [x,\gamma]$ is a function of x. By relaxing $\tilde{x}$ to 0 we obtain the lower bound as claimed. □

Now we are ready to handle (2.7) with high probability in the following proposition.

Proposition 2.6.19 (Concentration)**.** *For any fixed sequence $y_{1:n}$, and the corresponding language model's probability distribution p that gives conditional distributions $p_1, ..., p_n$. There exists a parameter $C_{\delta,\gamma}$ that depends only δ, γ. Then with probability at least $1-\beta$*

for any $\beta > 0$ (over G),

$$\sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim p(\cdot|x,y_{1:t-1})}[y_t \in G] \geq \frac{n\gamma e^{\delta}}{1+(e^{\delta}-1)\gamma} - C_{\delta,\gamma} \log^2 \frac{9(n+1)}{\beta} \left(\| \sum_{t=1}^{n} p_t[\cdot] \| + \sum_{t=1}^{n} \|p_t[\cdot]\|^2 + \| \sum_{t=1}^{n} p_t[\cdot] \|_\infty + \sum_{t=1}^{n} \|p_t[\cdot]\|_\infty^2 \right).$$

Proof. By Lemma 2.6.17 and 2.6.18

$$\begin{aligned}
&\sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim \hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G] \\
=&\sum_{t} \frac{e^{\delta} p_t(G)}{1+(e^{\delta}-1)p_t(G)} \\
\geq&\sum_{t} \frac{e^{\delta}\gamma}{1+(e^{\delta}-1)\gamma} + \frac{e^{\delta}(p_t(G)-\gamma)}{(1+(e^{\delta}-1)\gamma)^2} - e^{\delta}(p_t(G)-\gamma)^2 \\
=&\frac{n\gamma e^{\delta}}{1+(e^{\delta}-1)\gamma} + \frac{e^{\delta}}{(1+(e^{\delta}-1)\gamma)^2} \left(\underbrace{\sum_{t} \sum_{i=1}^{N\gamma} p_t[\pi[i]] - n\gamma}_{(*)} \right) - e^{\delta} \sum_{t} \left(\underbrace{\sum_{i=1}^{N\gamma} p_t[\pi[i]] - \gamma}_{(**)} \right)^2
\end{aligned}$$

where π is a random permutation of the index set $\{1, ..., N\}$.

We will now apply Lemma 2.6.28 to lowerbound $(*)$ with high probability and to bound the absolute value of $(**)$ with high probability.

Remark 2.6.20. The reason why we can apply these lemmas even after we condition on $y_{1:t-1}$ is due to the "high-probability homophily" assumption which allows us to use the fact that $y_{1:t-1}$ is independent to G, i.e., the distribution of the green list remains uniform at random after we condition on each qualifying $y_{1:t-1}$ separately.

Using a similar argument from the proof of Theorem 2.6.4, we can apply Lemma 2.6.28

and get that with probability $1-\beta$,

$$(*) \geq -\sqrt{64\gamma \| \sum_{t=1}^{n} p_t(\cdot)\|^2 \log(9/\beta)} - \| \sum_{t=1}^{n} p_t(\cdot)\|_\infty \log(9/\beta).$$

Similarly by Lemma 2.6.28 again to bound $(**) = \sum_{i=1}^{N\gamma} p_t[\pi[i]] - \gamma$ w.h.p for each t.

$$\big|(**)\big| \leq \sqrt{64\gamma \|p_t(\cdot)\|^2 \log(9/\beta)} + \|p_t(\cdot)\|_\infty \log(9/\beta).$$

To put things together, with probability $1-(n+1)\beta$,

$$\begin{aligned}
&\sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t \sim p(\cdot|x, y_{1:t-1})} [y_t \in G] \\
&\geq \frac{n\gamma e^\delta}{1+(e^\delta-1)\gamma} - \frac{e^\delta}{(1+(e^\delta-1)\gamma)^2} \left(\sqrt{64\gamma \| \sum_{t=1}^{n} p_t[\cdot]\|^2 \log(9/\beta)} + \| \sum_{t=1}^{n} p_t[\cdot]\|_\infty \log(9/\beta) \right) \\
&\quad - e^\delta \gamma(1-\gamma) \sum_t \|p_t[\cdot]\|^2 - 2e^\delta \left(64\gamma \sum_{t=1}^{n} \|p_t[\cdot]\|_2^2 \log(9/\beta) + \sum_{t=1}^{n} \|p_t[\cdot]\|_\infty^2 \log^2(9/\beta) \right) \\
&\geq \frac{n\gamma e^\delta}{1+(e^\delta-1)\gamma} - C_{\delta,\gamma} \log(9/\beta)^2 \left(\| \sum_{t=1}^{n} p_t[\cdot]\| + \sum_{t=1}^{n} \|p_t[\cdot]\|^2 + \| \sum_{t=1}^{n} p_t[\cdot]\|_\infty + \sum_{t=1}^{n} \|p_t[\cdot]\|_\infty^2 \right)
\end{aligned}$$

for a constant $C_{\delta,\gamma}$ that depends only in δ, γ. The proof is complete by defining $\tilde{\beta} = 9(n+1)\beta$, and get the same result under probability $1-\tilde{\beta}$. □

Many green list tokens in expectation

To obtain the lower bound in expectation, we just need to bound the expectation of (2.5), (2.6) and (2.7).

1. Observe that $\mathbb{E}[\text{Term (2.5)}|G] = 0$ (from Lemma 2.6.16)

2. Also, observe that (2.6) ≥ 0 under the *homophily* assumption (Assumption 2.6.11).

3. Term (2.7) can be further lower bounded by a second-order Taylor expansion argument (Lemma 2.6.18) and a variance calculation for sampling without replacement (Lemma 2.6.21), which ends up depending on the *on-average high-entropy* parameter from Definition 2.6.8. The formal result is stated in Proposition 2.6.22.

Lemma 2.6.21. *Fix p_t*

$$\mathop{\mathbb{E}}_{G}[(p_t(G)-\gamma)^2] \le \gamma(1-\gamma)\|p_t[\cdot]\|^2.$$

Proof. First observe that $\mathbb{E}_G[p_t(G)] = \gamma$ because every token has γ probability to be included. By the variance formula for sampling without replacement (N choose $N\gamma$),

$$\mathrm{Var}_G[p_t(G)|y_{1:t-1}] = \gamma N \frac{1}{N}\sum_{i=1}^{N}(p_t[i]^2 - N^{-2})(1-\frac{\gamma N-1}{N-1}) \le \gamma(1-\gamma)\sum_{i=1}^{N} p_t[i]^2.$$

□

Proposition 2.6.22. *Assume homophily, then*

$$\mathbb{E}\left[\sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t\sim\hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G]\right] \ge n\gamma\left(\frac{e^\delta}{1+(e^\delta-1)\gamma} - \frac{(1-\gamma)e^\delta}{n}\sum_{t=1}^{n}\mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)}\sum_{i=1}^{N} p_t[i]^2\right).$$

Proof. By homophily,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{n} \mathop{\mathbb{P}}_{y_t\sim\hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G]\right] \\
=&\sum_{t=1}^{n} \mathop{\mathbb{E}}_{G,y_{1:t-1}\sim\hat{p}(\cdot|x)}\left[\mathop{\mathbb{P}}_{y_t\sim\hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G]\right] \\
\ge&\sum_{t=1}^{n} \mathop{\mathbb{E}}_{G,y_{1:t-1}\sim p(\cdot|x)}\left[\mathop{\mathbb{P}}_{y_t\sim\hat{p}(\cdot|x,y_{1:t-1})}[y_t \in G]\right] \\
=&\sum_{t=1}^{n} \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \mathop{\mathbb{E}}_{G}\left[\frac{e^\delta \mathbb{P}_{y_t\sim p_t(\cdot|y_{1:t-1})}[y_t \in G]}{1+(e^\delta-1)\mathbb{P}_{y_t\sim p_t(\cdot|y_{1:t-1})}[y_t \in G]}\middle| y_{1:t-1}\right]
\end{aligned} \tag{2.8}$$

By Lemma 2.6.18, we can decompose (2.8). Also observe that $\mathbb{E}_G[p_t(G)|y_{1:t-1}] = \gamma$ where $p_t(G) := \mathbb{P}_{y_t \sim p_t(\cdot|y_{1:t-1})}[y_t \in G]$ is short hand for clarity. To see the second observation, notice that y_t is independent to G, thus we can apply Statement 1 of Theorem 2.6.4).

Apply the two observations to (2.8), we have

$$\begin{aligned}
(2.8) &\geq \sum_{t=1}^{n} \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \mathop{\mathbb{E}}_{G} \left[\frac{e^\delta \gamma}{1+(e^\delta-1)\gamma} + \frac{e^\delta(p_t(G)-\gamma)}{(1+(e^\delta-1)\gamma)^2} - e^\delta (p_t(G)-\gamma)^2 \middle| y_{1:t-1} \right] \\
&= \frac{e^\delta n\gamma}{1+(e^\delta-1)\gamma} + \sum_{t=1}^{n} \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \left[\frac{e^\delta(\mathbb{E}_G[p_t(G)|y_{1:t-1}]-\gamma)}{(1+(e^\delta-1)\gamma)^2} - e^\delta \mathop{\mathbb{E}}_{G}[(p_t(G)-\gamma)^2|y_{1:t-1}] \right] \\
&= \frac{e^\delta n\gamma}{1+(e^\delta-1)\gamma} - \sum_{t=1}^{n} e^\delta \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \mathrm{Var}_G[p_t(G)|y_{1:t-1}].
\end{aligned}$$

By the variance formula for sampling without replacement (N choose $N\gamma$),

$$\mathrm{Var}_G[p_t(G)|y_{1:t-1}] = \gamma N \frac{1}{N} \sum_{i=1}^{N} (p_t[i]^2 - N^{-2})(1 - \frac{\gamma N - 1}{N-1}) \leq \gamma(1-\gamma) \sum_{i=1}^{N} p_t[i]^2.$$

Thus, it follows that

$$\begin{aligned}
(2.8) &\geq \frac{e^\delta n\gamma}{1+(e^\delta-1)\gamma} - \sum_{t=1}^{n} e^\delta \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \gamma(1-\gamma) \sum_{i=1}^{N} p_t[i]^2 \\
&= n\gamma \left(\frac{e^\delta}{1+(e^\delta-1)\gamma} - \frac{(1-\gamma)e^\delta}{n} \sum_{t=1}^{n} \mathop{\mathbb{E}}_{y_{1:t-1}\sim p(\cdot|x)} \sum_{i=1}^{N} p_t[i]^2 \right).
\end{aligned}$$

□

2.6.7 Security property

Corollary 2.6.23. *Algorithm 2 with threshold τ satisfies the **security property** from Definition 2.3.2 with $\epsilon = 0$ and*

$$\eta(y, \mathsf{k}, \epsilon) = \frac{\sqrt{n}(z_y - \tau)}{1 + \gamma/2} \mathbf{1}\left(z_y - \tau \geq \frac{\gamma\sqrt{n}}{1 + \gamma/2}\right).$$

In comparison, the best bound on the security property parameter one can obtain for the scheme of [26] is (a formal statement and proof are included in Appendix 2.6.9)

$$\eta(y, \mathsf{k}, \epsilon) = \frac{\sqrt{n}(z_y - \tau)}{2 + \gamma/2} \mathbf{1}\left(z_y - \tau \geq \frac{\gamma\sqrt{n}}{2 + \gamma/2}\right).$$

To say it differently, our method, UNIGRAM-WATERMARK, utilizing a fixed Green-Red split, achieves *twice the robustness* to edits compared to [26]'s baseline approach.

Analysis of KGW Watermark

2.6.8 Soft watermarking scheme of [26]

This section illustrates the soft watermarking scheme proposed by [26]. This straightforward algorithm only requires access to the language model's logits at each time step. Let $y = [y_1, \ldots, y_n]$ represent the output sentence of language model $\mathcal{M}$ given the prompt x. The watermarking scheme generates $y_{1:n}$ by hashing y_{t-1} to a partition of the token space (Green List and Red List) and amplifies the probability of tokens on the Green List. Specifically, $[y_1, \ldots, y_n]$ is derived from the following Markov chain:

1. $y_1 \sim \text{Softmax}\big(\text{logits}_{\mathcal{M}}(y_1 = \cdot|x)\big)$

2. For $t = 2 : n$,

$$y_t \sim \text{Softmax}\big(\text{logits}_{\mathcal{M}}(y_t = \cdot|[x, y_1 \ldots, y_{t-1}]) + \delta \mathbf{1}(\cdot \in \text{Green}(y_{t-1}))\big)$$

Typically, $\gamma|\mathcal{V}|$ tokens are selected to form a Green List, where γ symbolizes the fraction of tokens to be watermarked (by default, $\gamma = 0.5$). The logit value for each green token is augmented by a constant δ (default value = 2), which denotes the watermark strength. This elevation enhances the likelihood of sampling green, watermarked tokens, particularly for high-entropy distributions.

Validation of whether a text was generated by a watermarked language model is achievable given knowledge of the hash function and tokenizer. The adversary constructs $u = [u_1, \ldots, u_m]$ from $x, y_{1:n}$ and any auxiliary input. The detection algorithm calculates the quantity of green tokens $|u|_G = \sum_{t=2}^{m} \mathbf{1}(u_t \in \text{Green}(u_{t-1}))$. One can assume the null hypothesis, denoted as H_0*: The text sequence is produced independently of the green list rule.* Following this, a z-statistic score is computed as $z = (|u|_G - \gamma m) / \sqrt{m\gamma(1-\gamma)}$. If the z-score exceeds a predetermined threshold, the algorithm declares, "This was generated from $\hat{\mathcal{M}}$!".

2.6.9 Security property of [26]

We also demonstrate the robustness property of the soft watermarking algorithm in [26] in the following Theorem 2.6.24

Theorem 2.6.24 (Robustness to editing in the watermarking scheme of [26])**.** *Let $y = [y_1, \ldots, y_n]$ represent the watermarked sequence. Suppose the adversary $\mathcal{A}$ follows the definition 2.3.2 and outputs a modified text $u = [u_1, \ldots, u_m]$. Following Equation 2.2, we*

calculate the z-score of the soft watermarking [26] z_y and z_u. Then we have

$$z_u \geq z_y - \max\{\frac{(2+\gamma/2)\eta}{\sqrt{n}}, \frac{(2-\gamma/2)\eta}{\sqrt{n-\eta}}\}.$$

Proof. The proof is similar to that of Theorem 2.3.10 except that the maximum perturbation to $|\mathbf{y}|_G$ is now 2η rather than η. We now justify that the maximum perturbation has really doubled below, but ignore the part that is the same as in the proof of Theorem 2.3.10.

Let $\mathsf{BiGrams}(u) = \{\{u_1, u_2\}, \{u_2, u_3\}, ..., \{u_{n-1}, u_n\}\}$ and similarly $\mathsf{BiGrams}(y)$ enumerates the set of all two grams in sequence $y_{1:m}$.

We claim that each edit can modify at most two elements in the above set. To see this, consider "insertion", "deletion", and "edit" separately.

- If we "insert" one token $\tilde{u}$ at t, then $\{u_{t-1}, u_t\}$ and $\{u_t, u_{t+1}\}$ become $\{u_{t-1}, \tilde{u}\}$, $\{\tilde{u}, u_t\}$ and $\{u_t, u_{t+1}\}$. Only one element of $\mathsf{BiGrams}(u)$ is modified — $\{u_{t-1}, u_t\}$.
- For "deletion" at t, $\{u_{t-1}, u_t\}$ and $\{u_t, u_{t+1}\}$ become $\{u_{t-1}, u_{t+1}\}$. So two elements from $\mathsf{BiGrams}(u)$ are gone.
- For "edit" at t, $\{u_{t-1}, u_t\}$ and $\{u_t, u_{t+1}\}$ become $\{u_{t-1}, \tilde{u}\}$ and $\{\tilde{u}, u_{t+1}\}$. Thus again only two elements from $\mathsf{BiGrams}(u)$ are gone.

It follows that when y is obtained after up to η edits

$$|\mathsf{BiGrams}(u) \cap \mathsf{BiGrams}(y)| \geq |\mathsf{BiGrams}(u)| - 2\eta$$

Observe that $\sum_{t=2}^{n} \mathbf{1}(u_t \in \text{Green}(u_{t-1}))$ counts the number of qualifying elements in $\mathsf{BiGrams}(u)$, which completes the proof. □

For this reason, our watermark is twice as robust as that of [26]. This provides the theoretical guarantee to our empirical results presented in the experiments!

Remark 2.6.25. We can view our watermark as a trivial Markovian watermarking scheme with $k = 0$, and what [26] proposed to be $k = 1$. For the more general k-Markovian watermarking scheme that depends on a prefix of length k, the robustness deteriorates by a factor of k, as the maximum perturbation will become $\frac{((k+1)+\gamma/2)\eta}{\sqrt{n}}$. To say it differently, choosing $k = 0$ gives the maximum robustness and maximum simplicity at the same time, and the benefit leads to significant gains in our experiments, especially against paraphrasing attacks.

Alternative detector "Unique" and its desirable properties

Our theoretical analysis suggests a promising alternative Detect algorithm for UNIGRAM-WATERMARK that simply involves calling Algorithm 2 with a deduplicated y.

Algorithm 4 UNIGRAM-WATERMARK: Detect (Alternative)

1: **Input:** suspect text y, watermark detection key k, threshold τ.
2: **Output:** 1 or 0 (whether the text is watermarked).
3: **Return** Algorithm 2 with suspect text Unique(y), detection key k and threshold τ.

The simple change actually results in a number of interesting new properties. For example, we can state its Type I error bound a lot more cleanly now as a Corollary of Theorem 2.6.4

Corollary 2.6.26 (No false positive for Deduplicated Detection)**.** *Consider $y = y_{1:n}$ as any* fixed *suspect text. Let $m = |\text{Unique}(y)|$ be the number of unique tokens in y. Let*

G be selected through Algorithm 1, using a uniform random choice. Then the following statements hold true:

1. *Assume $m \geq 1$, then*

$$\mathbb{E}[|\text{Unique}(y)|_G|y] = \gamma n \quad \textit{and} \quad \mathbb{E}[z_{\text{Unique}(y)}|y] = 0.$$

2. *With probability $1-\alpha$ (over only the randomness of G),*

$$\mathbb{P}\left[|\text{Unique}(y)|_G \geq \gamma m + \sqrt{64\gamma m \log(9/\alpha)} + \log(9/\alpha)\middle| y\right] \leq \alpha$$

or equivalently (when $n \geq 1$)

$$\mathbb{P}\left[z_{\text{Unique}(y)} \geq \sqrt{\frac{64\log(9/\alpha)}{(1-\gamma)}} + \frac{\log(9/\alpha)}{\sqrt{m\gamma(1-\gamma)}}\middle| y\right] \leq \alpha.$$

The above gives a clean finite-sample concentration bound of the Type I error using Algorithm 4. Notably, while deduplicating reduces the length of the suspect text, i.e., $m < n$, it improves the bound by ensuring both $C_{\max}$ and V are 1.

Remark 2.6.27 (Asymptotic choice of τ for controlling false positives). Lemma 2.6.21 gives that

$$\text{Var}\left[|\text{Unique}(y)|_G|y\right] = m\gamma(1-\gamma)(1-\frac{m-1}{N-1})$$

i.e., the conditional variance of $z_{\text{Unique}(y)}$ is $(1-\frac{m-1}{N-1})$. This means that if we want to control the asymptotic false positive rate to α, all we have to do is to choose the threshold τ to be

$$\tau = \sqrt{1-\frac{m-1}{N-1}}\Phi^{-1}(1-\alpha) \tag{2.9}$$

where Φ is the standard normal CDF.

Type II error. How about Type II error? Our results in Theorem 2.6.13 are still applicable but require us to apply that with a special language model derived from the original that directly generates Unique(y) (ordered in the same order they appear in y). This is still a valid autoregressive language model but has different roll-out probabilities.

Robustness to Edits. Observe that adding/removing/replacing one token to y in the results in adding/removing/replacing one token to Unique(y) respectively, the robustness of the z-score for Unique(y) thus directly follows Theorem 2.6.24.

"Unique" in K-gram watermark section with $K \geq 2$. Clearly, the same idea of deduplication works for the whole family of K-gram watermark proposed in [26]. In fact, it was briefly mentioned in a remark from their paper as a mitigation measure to reduce correlation. All arguments we make about Type I error and Robustness to Edits above work for $K \geq 2$. We defer the Type II error bound for this family to future work.

Emperical analysis on controlling false positives. We conduct experiments to demonstrate the results for the asymptotic choice of τ in controlling false positives. The negative examples are sampled from diverse datasets, including human data in LFQA and OpenGen dataset [45], C4 dataset [54], and TOEFL dataset [25]. In total, we collect 6,200 unwatermarked text samples with varied lengths. We then use the dynamic threshold τ with different choices of α as shown in Equation 2.9. By choosing different random seeds, we obtain different green lists. The results in Figure 2.8 show the empirical false positive rate aligns well with the theoretical α.

2.6.10 Alternative detection "Unique" is robust to Emoji attack and other tricky attacks

[26] discussed a number of interesting attacks on the K-gram watermarks. In this section, we inspect the robustness of UNIGRAM-WATERMARK (with both Algorithm 2

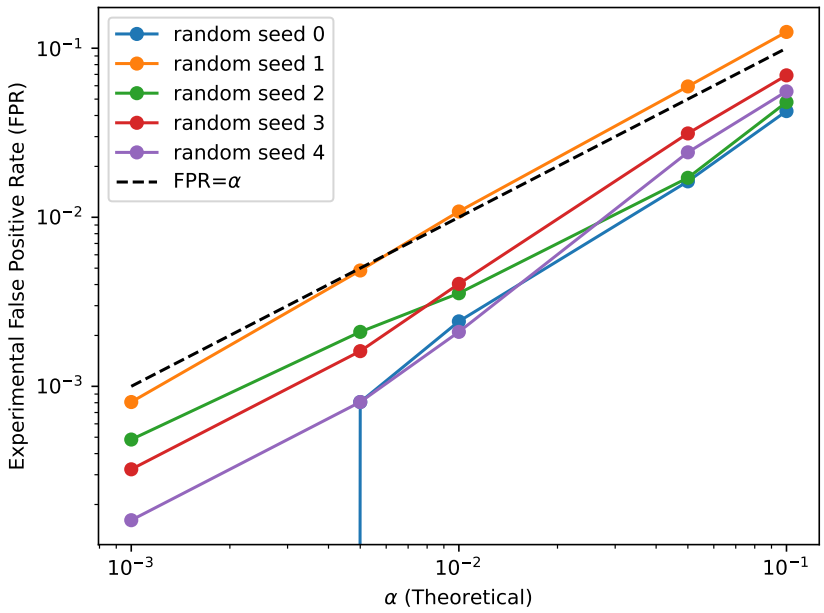

Figure 2.8: Empirical vs. theoretical false positive rates across various α values, using multiple green list initializations.

and 4 as Detect) to these attacks.

We will focus on those trickier generative attacks, as those non-generative attacks on the surface level (e.g., synonym substitution, Unicode substitution) were rather satisfactorily addressed in [26]. The same arguments work for UNIGRAM-WATERMARK. However, there are trickier ones that break K-gram watermarks for $K \geq 2$ but not for $K = 1$, especially when we use Algorithm 4 for detection.

Emoji attack the Emoji attack, also known as the Pineapple attack, asks the language model to inject a special symbol, e.g., an Emoji, in between the actual text that the LM is supposed to generate in response to a prompt. For example, a user of the language model can prompt an LM with "`Write my college admission essay. Insert an emoji in between every word.`". Then the user can simply remove the artificially injected symbol before submitting the essay. Clearly, this attack breaks all K-gram watermarks for $K > 1$ [26] (but also [27]). Our UNIGRAM-

WATERMARK remains effective because half of the tokens in y are still watermarked. The repeated Emoji symbol with length $n/2$ is a problem for both Type I and Type II error, but after "Unique", this corresponds to just an Edit Distance of 1 on Unique(y)!

Alphabet attack We can also make the Emoji attack stronger by injecting a known sequence of "alphabets" instead of one single symbol. For example, "`Write my college admission essay. Insert, in the order of the vocabulary, a token in between every word you generate.`". The alphabet attack breaks K-gram watermarks in the same way an Emoji attack does, but since the added tokens are different, "Unique" does not solve it right away. Interestingly, despite $m/2$ of the tokens in Unique(y) are not watermarked, as long as the Alphabet being used is independent of the secret key, it does not change the Type I error and only slightly reduces the power (i.e., 1-Type II error) since the expected number of Green tokens in that $m/2$ injected tokens is $m\gamma/2$.

Stegnography attack One may extend the attack even further by asking the language model to encode a message, which swaps each token in the vocabulary with another token through a secret codebook. For example, whenever you want to output Token i, output Token mod$(i+1, N)$ instead. If the "code book" is supplied in the prompt with an instruction for the LM to follow the code book when generating the text, then it really breaks all watermarks including ours, while allowing the user who knows the code book to easily revert it to the original text. The issue of such an attack is that it requires significantly heavy-lifting for the language model to predict outside the typical distribution it is trained on. There is no real risk of such an attack being employed as it is likely to significantly reduce the quality of the generated text.

To be clear, these attacks are, in fact, not post-processing-based evasion attacks, but rather hacks into prompts. Nevertheless, our watermark that is robust to edits turns out to be quite resilient to them.

Technical lemmas

Lemma 2.6.28 (Bernstein-style inequality for random permutation [29, Proposition 2.2]). *Let $\{a_{i,j}\}_{1\le i,j\le n}$ be a collection of non-negative numbers and Π_n be a random uniform permutation. Let $Z_n = \sum_{i=1}^n a_{i,\Pi_n(i)}$. Then, for any $t > 0$*

$$\mathbb{P}\left[|Z_n - \mathbb{E}[Z_n]| \ge 2\sqrt{\frac{t}{n}\sum_{i,j=1}^n a_{i,j}^2} + \max_{1\le i,j\le n}\{a_{i,j}\}t\right] \le 8e^{1/16}e^{-\frac{t}{16}}.$$

Lemma 2.6.29 (Variance for sampling without replacement). *Let $x_1, ..., x_N \in \mathbb{R}$. For any sample size $1 \le n \le N$, and π be a random permutation of $\{1, 2, ..., N\}$. The variance of $X = \frac{1}{n}\sum_{i=1}^n x_{\pi(i)}$ satisfies*

$$\mathrm{Var}(X) = \frac{1}{nN}\sum_{i=1}^N (x_i - \bar{x})^2(1 - \frac{n-1}{N-1}).$$

Definition 2.6.30 (Martingale). A sequence of random variables $(X_n)_{n\in\mathbb{N}}$ is called a *martingale* if it satisfies the following conditions:

1. $\mathbb{E}[|X_n|] < \infty$ for all $n \in \mathbb{N}$.
2. $\mathbb{E}[X_{n+1}|\mathcal{F}_n] = X_n$ for all $n \in \mathbb{N}$.

where $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq ... \subseteq \mathcal{F}_n \subseteq \mathcal{F}_{n+1} \subseteq ...$ is a filtration. Specifically, $\mathcal{F}_n$ can be the sigma-algebra generated by another sequence of random variable $Y_1, ..., Y_n$, i.e., $\mathcal{F}_n = \sigma(Y_{1:n})$ and X_n can be a function of $Y_{1:n}$.

Lemma 2.6.31 (Azuma-Hoeffding Inequality)**.** *Let $(X_n)_{n\in\mathbb{N}}$ be a martingale such that $|X_{n+1} - X_n| \leq c_n$ for some constants c_n and all $n \in \mathbb{N}$. Then for all $t > 0$ and $n \in \mathbb{N}$, we have*

$$\mathbb{P}\left(|X_n - X_0| \geq t\right) \leq 2\exp\left(-\frac{t^2}{2\sum_{i=1}^{n} c_i^2}\right).$$

Chapter 3

PF-Watermark

In this chapter, we propose a new decoding method called Permute-and-Flip (PF) decoder. It enjoys robustness properties similar to the standard sampling decoder, but is provably up to 2x better in its quality-robustness tradeoff than sampling and never worse than any other decoder. We also design a cryptographic watermarking scheme analogous to [27]'s Gumbel watermark, but naturally tailored for PF decoder. The watermarking scheme does not change the distribution to sample, while allowing arbitrarily low false positive rate and high recall whenever the generated text has high entropy. Our experiments show that the PF decoder (and its watermarked counterpart) significantly outperform(s) naive sampling (and it's Gumbel watermarked counterpart) in terms of perplexity, while retaining the same robustness (and detectability), hence making it a promising new approach for LLM decoding. Our code is available at https://github.com/XuandongZhao/pf-decoding.

3.1 Introduction

Large language models (LLMs) [1, 14, 57, 2] have become increasingly popular in recent years due to their ability to generate human-like text and solve many tasks through a natural chatbot interface.

A language model predicts the next word in a sentence using a real-value function $u(\cdot; \text{prompt}, \text{prefix}) : \mathcal{V} \to \mathbb{R}$, known as *logits*, which encodes the model's preferences on which word to choose. Here $\mathcal{V}$ is the vocabulary space (typically a large discrete set of words); the "prompt" describes the task of interest; and "prefix" includes all preceding words that have been generated so far.

A language model *decoder* refers to a possibly randomized function that takes a prompt text x, API access to the *logits* function as input, and outputs a sentence $y_{1:n}$.

The main thrust of this chapter is to introduce a new decoder, termed *Permute-and-Flip decoding*, work out some of its intriguing properties with an application to watermarking LLM text, and hopefully convince readers that it deserves a shot at your next LLM application.

3.1.1 Problem Setup and Summary of Results

Before we get into it, let us set up the stage with a quick tour to the zoo of existing decoding methods and have a brief sneak-peek into the "jar of worms" on how a language model decoder can be evaluated.

Popular existing decoding methods fall into three categories: (1) Planning-based methods such as beam search that aims at maximizing the sequence likelihood; (2) sampling-based methods that recursively sample from the next-word distribution, e.g.,

the soft(arg)max transform of the logits

$$\textbf{Softmax sampling: } y_t \sim p(y) = \frac{e^{u(y|x,y_{1:t-1})/T}}{\sum_{\tilde{y}} e^{u(\tilde{y}|x,y_{1:t-1})/T}} \tag{3.1}$$

where T is the *temperature* parameter; and (3) greedy methods such as greedy decoding that simply outputs $y_t = \arg\max_{y \in \mathcal{V}} u(y|x, y_{1:t-1})$ as well as its Top p [58] and Top k sampling variants that interpolate greedy and sampling methods.

Performance metrics. How do we compare different decoding methods? More generally, how do we evaluate LLM-generated text? These are questions far from being settled. Naturally, if there is a (possibly task-dependent) performance metric $U_x : \mathcal{V}^n \to \mathbb{R}$ one can define, then the optimal decoder would be the one that outputs

$$y^*_{1:n} = \arg\max_{y_{1:n} \in \mathcal{V}^n} U_x(y_{1:n}).$$

Often U_x is instantiated to be the sequence likelihood $\sum_{t=1}^{n} \log p(y_t|x, y_{1:t-1})$ which is equal to $\sum_{t=1}^{n} u_t(y_t)$.

Recent works [59, 60], however, report that strategies that aim at maximizing sequence likelihood often result in texts that are more repetitive and less effective in some downstream tasks than those from the sampling-based methods [58]. Depending on what the task is, there is not a one-size-fits-all performance metric, therefore is no single decoding method that works well for all tasks.

For the moment, let us stash the disputes on how to best evaluate an LLM-generated text and focus on designing methods that maximize any user-specified utility function. In fact, we will also give up on solving the sequence-level utility maximization problem[1] and simply maximize a *per-step* utility function $u_t : \mathcal{V} \to \mathbb{R}$.

[1] It is known to be NP-Complete [61].

u_t can simply be the logits function that LLMs output, which may have already accounted for potential future utility (like the Q function in reinforcement learning) since the transformer-based language model had access to future texts during pre-training. Or u_t can be explicitly augmented with structure-inducing regularizers such as a lookahead heuristic as in A* decoding [62], a retrieval-based term for fact-checking [63], or an entropy bonus for promoting diversity [64].

Our goal is thus to construct a possibly randomized algorithm $\mathcal{A}$ that takes u_t as an input and outputs $y_t \in \mathcal{V}$ that aims at maximizing $\mathbb{E}_{y_t \sim \mathcal{A}_{u_t}}[u_t(y_t)]$ as much as possible. In the remainder of the chapter, we will simply take u_t as "logits" for a concrete exposition — all results are valid when u_t is instantiated otherwise.

Other constraints / consideration. Why doesn't the trivial greedy decoder work? That's because there are other considerations besides text quality when selecting LLM decoders. For example, **computational efficiency and latency** are hugely important, since each API call to the *logits* function is costly. The **diversity** of the generated text is also important, especially for creative tasks.

Moreover, the decoding procedure should be **watermarkable** [27, 26, 9, 65] in the sense that one should be able to inject subtle statistical signals that can be retrieved when given a secret key, to *prove* that the text is generated by this particular language model. Being watermarkable prevents the LLM from being used for malicious purposes such as scams [16], fake news [15], and plagiarism [17].

In addition to the above, one may also hope the decoding algorithm to be **robust against small perturbations** to the *logits*. Specifically,

Definition 3.1.1 (Robustness)**.** We say a decoding algorithm $\mathcal{A}$ is L-robust if for any prompt x, prefix $y_{<=t}$, and for any perturbed $\tilde{u}$ such that $\|\tilde{u} - u\|_\infty \leq \delta$, the log-probability

Methods	Perplexity	Computational Efficiency	Diversity	Watermark	Robustness
Search (e.g., Beam)	Lowest	✗	✗	✗	✗
Greedy	Low	✓	✗	✗	✗
Softmax Sampling	Moderate	✓	✓	✓	✓
Top-p Sampling	Low (for small p)	✓	Depends on p	✓	✗
Top-k Sampling	Low (for small k)	✓	Depends on k	✓	✗
PF Sampling (ours)	Lower than Softmax	✓	✓	✓	✓

Table 3.1: Comparison of different decoding methods against five desiderata.

ratio satisfies

$$\left|\log\left\{\frac{p_{\mathcal{A}(\tilde{u}(\cdot|x,y_{<=t}))}(y)}{p_{\mathcal{A}(u(\cdot|x,y_{<=t}))}(y)}\right\}\right| \leq L\delta \quad \forall y \in \mathcal{V}.$$

The robustness helps to avoid catastrophic failure in the scenarios where the logits may be subject to data poisoning [66, 67] or jailbreaking attacks [68, 69].

Robustness implies an intuitive notion of *diversity*, which says that for tokens with similar logits, then their chances of getting chosen should be similar. More rigorously:

Remark 3.1.2 (Robustness implies Diversity). If $|u_t(y)-u_t(y')| \leq \delta$, then we can construct a $\tilde{u}_t$ such that $\tilde{u}_t(y) = \tilde{u}_t(y')$ while satisfying $\|u_t - \tilde{u}_t\|_\infty \leq \frac{\delta}{2}$. Apply triangle inequality and Definition 3.1.1, we get

$$\left|\log\frac{p_{\mathcal{A}_{u_t}(y)}}{p_{\mathcal{A}_{u_t}(y')}}\right| = \left|\log\frac{p_{\mathcal{A}_{u_t}(y)}}{p_{\mathcal{A}_{\tilde{u}_t}(y)}} + \log\frac{p_{\mathcal{A}_{\tilde{u}_t}(y')}}{p_{\mathcal{A}_{u_t}(y')}}\right| \leq L\delta.$$

Inspecting the decoding methods along the aforementioned dimensions, we notice that planning-based methods fail to be computationally efficient. While greedy decoding is efficient and has relatively low perplexity, its generated texts are neither diverse nor watermarkable (at least not using existing techniques). The sampling-based methods, however, are both watermarkable and diverse. In addition, softmax sampling is known to be 2-robust, while all other methods that we discussed so far are not robust.

Fact 3.1.3. *Softmax sampling decoding using* (3.1) *with temperature* T *satisfies* L*-robustness*

with $L = 2/T$.

Proof. The result is implied by the differential privacy guarantee of exponential mechanism [55, Theorem 6]. □

The pros and cons of different decoding methods are summarized in Table 3.1. From the table, we can see that there is a clear tradeoff between minimizing perplexity and preserving other properties. In particular, softmax sampling is the only method that checks all boxes, and the only one that is robust among existing decoders. This observation begs the following research question:

Is there a decoding method that is as robust as softmax sampling, but has lower perplexity?

In this chapter, we answer this question affirmatively by bringing in a technique called Permute-and-Flip sampling.

Our contributions are fourfold.

1. We introduce Permute-and-Flip decoding — a new decoding algorithm for language models based on recent development in a very different context [70].
2. We demonstrate that existing results from [70] already imply that:
   - Permute-and-Flip decoding is provably robust.
   - The robustness-perplexity tradeoff of the PF decoding is Pareto-optimal. In particular, when compared to softmax sampling, PF decoding has up to 2x smaller expected suboptimality while having the same robustness parameter L.
3. We designed an analog of [27]'s Gumbel-Watermark for PF decoder, called the PF watermark. We show that the watermarked PF decoder samples from a distribution that is computationally indistinguishable from the non-watermarked PF decoder, and

the detection procedure has precisely controlled false positive rate (FPR) and high power in identifying watermarked texts.

4. We empirically demonstrate that on open-generation tasks, PF watermark achieves the best balance of the highest detection accuracy and lowest perplexity compared to the baselines.

Overall, our proposed permute-and-flip decoding method provides a promising approach to balancing the tradeoff between perplexity and robustness in LLM decoding while also admitting watermarking capabilities.

Related work and novelty. PF sampling was invented in the differential privacy (DP) literature [70]. Its robustness properties are well-understood. The robustness of Softmax sampling is also well-known [55]. Our contribution is in applying this method to LLM decoding and connecting these known theoretical results to the broader ML audience. To our knowledge, the PF watermark is new to this chapter. The design of the watermark leverages the Report-Noisy-Max interpretation of the PF sampling [71] which allows a similar pseudo-random function like the work of [27] to be applied.

A more thorough discussion of the related work is given in Appendix 3.6.2.

3.2 Permute-and-Flip Decoding its Properties

The Permute-and-Flip decoding iteratively generates the next token by a simple procedure that uses only the logits. It involves first randomly permuting the vocabulary, then flipping a sequence of biased coins according to the permuted sequence until the first "head" is seen (see Algorithm 5).

Permute-and-flip makes words with higher logits exponentially more likely — even more so than Softmax sampling (3.1). To see this, one may consider a rejection sampling

Algorithm 5 Permute and Flip (PF) Decoding

Input: prompt x, language model $\mathcal{M}$, temperature T.
for $t = 1, 2, \cdots$ **do**
  Logits $u_t \leftarrow \mathcal{M}([x, y_{1:t-1}])$.
  Find $u_t^* \leftarrow \max_{y \in \mathcal{V}} u_t(y)$.
  Permute : Shuffle the vocabulary $\mathcal{V}$ into $\tilde{\mathcal{V}}$.
  for $y \in \tilde{\mathcal{V}}$ **do**
    Flip : Draw $Z \sim \text{Bernoulli}\left(\exp\left(\frac{u_t(y)-u_t^*}{T}\right)\right)$.
    if $Z = 1$, **then** assign $y_t \leftarrow y$ and **break**.
  end for
end for
Output: Generated sequence $y = [y_1, ..., y_n]$.

algorithm for obtaining a sample from (3.1), which repeats the following procedures until it halts.

1. Uniformly samples $y \in \mathcal{V}$,

2. Return it with probability

$$p(y)/p(y^*) = \exp\left((u_t(y) - u_t(y*))/T\right).$$

The only difference from the PF sampling is that this procedure samples y *with replacement* in every iteration, while the PF sampling samples y *without replacement*. Intuitively, PF sampling has a higher chance of outputting y^*.

PF sampling was initially proposed in [70] as a differentially private selection mechanism that has better utility than the more well-known exponential mechanism [55]. [70] also derived a plethora of theoretical properties of the PF sampling. The following theorem summarizes these results in the language of LLM decoding.

Theorem 3.2.1. *Let the logits function be* u *and* $u^* = \max_{y \in \mathcal{V}} u(y)$. *Let* $\text{PF}(u)$ *be the distribution of PF-sampling, and* $\text{Softmax}(u)$ *be the distribution in* (3.1)*, both with temperature parameter* T. *The following statements are true.*

1. *(**Same robustness**) PF-Sampling is $(2/T)$-robust.*

2. *(**Nearly greedy**) PF-sampling obeys*

$$\mathop{\mathbb{E}}_{y\sim\mathrm{PF}(u)}[u(y)] \geq u^* - T\log|\mathcal{V}|.$$

3. *(**"Never worse"**) For the same T, PF-sampling is never worse than Softmax-sampling.*

$$\mathop{\mathbb{E}}_{y\sim\mathrm{PF}(u)}[u(y)] \geq \mathop{\mathbb{E}}_{y\sim\mathrm{Softmax}(u)}[u(y)]$$

4. *(**"Up to 2x better"**) There exists logits u such that PF-sampling is 2x smaller in terms of suboptimality*

$$\mathop{\mathbb{E}}_{y\sim\mathrm{PF}(u)}[u^* - u(y)] \leq \frac{1}{2}\mathop{\mathbb{E}}_{y\sim\mathrm{Softmax}(u)}[u^* - u(y)].$$

5. *(**Optimal robustness-perplexity tradeoff**) For any decoder P that is $2/T$-robust, if there exists u such that*

$$\mathop{\mathbb{E}}_{y\sim P(u)}[u(y)] > \mathop{\mathbb{E}}_{y\sim\mathrm{PF}(u)}[u(y)]$$

then there must be another $\tilde{u}$ such that

$$\mathop{\mathbb{E}}_{y\sim P(\tilde{u})}[\tilde{u}(y)] < \mathop{\mathbb{E}}_{y\sim\mathrm{PF}(\tilde{u})}[\tilde{u}(y)].$$

Proof. The theorem is entirely due to [70]. The five statements are directly implied by Theorem 1, Corollary 1, Theorem 2, Proposition 4, and Proposition 6 respectively in their paper. □

The first statement shows that the PF decoder enjoys exactly the same robustness parameter as in Fact 3.1.3. The second statement provides a worst-case bound on how far PF-sampling is away from greedy-decoding as a function of the temperature T in terms of the likelihood achieved. The third and fourth statements show that PF-sampling is always "more greedy" than softmax-sampling. The last statement shows that PF-sampling is not dominated by any other decoder that is equally robust (as in Definition 3.1.1), thus *Pareto optimal*.

These results provide strong justification on the superiority of the permute-and-flip decoder over the standard softmax sampling in minimizing perplexity.

Let's consider a simple example to compare PF decoder and Softmax decoder.

Example 3.2.2. Let the $|\mathcal{V}| = 2$ and the corresponding logits be $[\Delta, 0]$ for gap $\Delta > 0$. Softmax decoder chooses the suboptimal token with probability $1/(1 + e^{\Delta/T})$, while PF decoder chooses it with probability $\frac{1}{2}e^{-\Delta/T}$.

Since $1/(1 + x) > 1/(2x)$ for all $x > 1$, the probability that the suboptimal token is chosen in PF sampling is strict smaller than that of Softmax sampling (also see Figure 3.1).

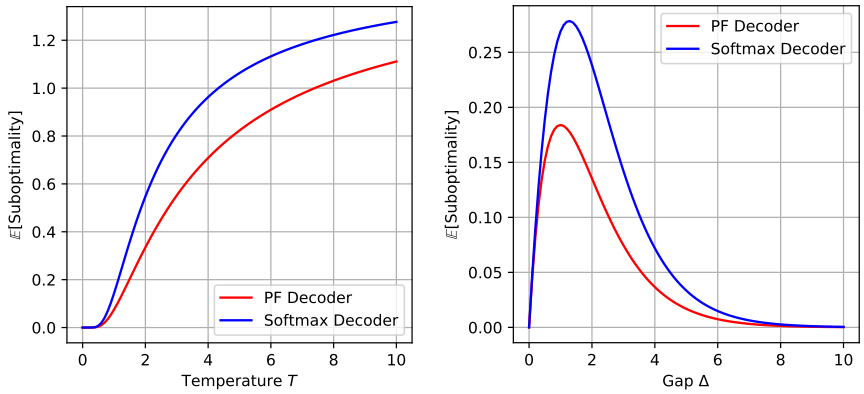

Figure 3.1: Comparing PF decoder vs Softmax decoder using Example 3.2.2. On the left, we fix the Gap $\Delta = 3.0$ and vary the temperature T. On the right, we fix $T = 1.0$ and consider vary Δ. PF beats Softmax in all cases.

3.3 Report-Noisy-Max and Watermarking

Next we turn to the well-motivated problem of watermarking LLM generated text. The watermarking problem aims at embedding a secret message in the generated text that (essentially) reads "Beware! I am written by an AI!".

The hope is that this message can be seen by anyone who has access to a *secret key*, while ensuring that the watermarked version of the LLM generates text that has *almost the same* distribution as (or at least very similar) to the original LLM.

More formally, a watermarking scheme includes a "Watermark" function that injects the watermark and a "Detect" function that takes a suspect text sequence $y_{1:n}$ as input and outputs a prediction of 1 ("It is watermarked!") or 0 ("It is not!").

A wrong accusation of non-watermarked text as watermarked is called a *false positive*. A failure to detect a watermarked text is called a *false negative*. The performance of a watermark is measured by its detection *power* (i.e., 1−*false negative rate*) at a given *false positive rate*.

There are many other necessary properties for a watermarking scheme to be useful, such as *low-overhead*, *model-agnostic* detection, and *resilience to edits* and other evasion attacks. We refer readers to the slide deck of [27] and the related work section of [6] for a review of these desiderata and known results.

Among the recent attempts, two popular watermarking schemes perform satisfactorily on all the above criteria.

Gumbel Watermark [27] that uses a "traceable" pseudo-random softmax sampling when generating the next word.

Green-Red Watermark [26] that randomly splits the vocabulary into Green and Red then slightly increases the logits for green tokens.

Both of them determine their pseudo-random seeds chosen according to the m preceding

tokens of the current token being generated. We will focus on explaining the Gumbel watermark as it is more closely related to our approach.

[27]'s Gumbel watermark. The key idea of the Gumbel watermark leverages the "Gumbel-Max Trick", which states that:

Fact 3.3.1 ([72]). *The softmax sampling in* (3.1) *is equivalent to the following procedure*

$$y_t = \arg\max_{y \in \mathcal{V}} \frac{u_t(y)}{T} + G_t(y) \tag{3.2}$$

where $G_t(y) \sim$ *Gumbel*$(0,1)$ *i.i.d for each* t, y.

Gumbel noise can be generated using a uniform r.v..

$$\text{Gumbel}(0,1) \sim -\log\left(\log(1/\text{Uniform}([0,1]))\right).$$

So given a random vector $r_t \sim (\text{Uniform}([0,1]))^{|\mathcal{V}|}$, we can write $G_t(y) = -\log(-\log(r_t(y)))$.

The Watermark stage for the Gumbel-watermark essentially replaces Uniform$([0,1])$ with a *pseudo-random function* $r_t(y) = F_{y_{t-m:t-1},\mathsf{k}}(y)$. Given the secret key k, the *pseudo-random function* is a deterministic function with range $[0,1]^{\mathcal{V}}$, but over the distribution of the secret key k, r_t is computationally indistinguishable from sampled from truly i.i.d. uniform distribution, which ensures that the distribution of y_t in the watermarked model is computationally indistinguishable to the unwatermarked distribution (3.1).

At Detect phase of the the Gumbel watermark, the auditor who has access to the key k may compute

$$\text{TestScore}_{\text{Gumbel}}(y_{1:n}) = \sum_{t=m+1}^{n} -\log(1 - r_t(y_t)).$$

If $y_{1:n}$ is *not* generated from the watermarked model, then the test statistic is a sum of exponential random variable thus $\mathbb{E}[\text{TestScore}(y_{1:n})] = n - m$.

Meanwhile, it was shown by [27] that if $y_{1:n}$ is generated by the Gumbel watermarked model,

$$\mathbb{E}[\text{TestScore}(y_{1:n})] = \sum_{t=m+1}^{n} \mathbb{E}\left[\sum_{y\in\mathcal{V}} p_t(y) H_{\frac{1}{p_t(y)}}\right] \tag{3.3}$$

$$\geq (n-m) + \left(\frac{\pi^2}{6} - 1\right) \sum_{t=m+1}^{n} \mathbb{E}\left[\text{Entropy}[p_t(\cdot)]\right]. \tag{3.4}$$

where $p_t := \text{Softmax}(u_t)$, $H_\alpha := \int_0^\alpha \frac{1-x^\alpha}{1-x} dx$ is Euler's Harmonic number and Entropy denotes the standard Shannon entropy (in nats) for a discrete distribution, i.e., $\text{Entropy}[p] = -\sum_{y\in\mathcal{V}} p(y) \log p(y)$.

Permute-and-Flip as ReportNoisyMax. It turns out that the Permute-and-Flip sampling has a similar equivalent Report-Noisy-Max form. Instead of Gumbel noise, it is the exponential noise that are added to the logits. This less-known fact is due to [71]

Fact 3.3.2 ([71, Theorem 5])**.** *Permute-and-Flip Sampling in Algorithm 5 with parameter T is equivalent to*

$$y_t = \arg\max_{y\in\mathcal{V}} \frac{u_t(y)}{T} + E_t(y). \tag{3.5}$$

where $E_t(y) \sim$ Exponential(1) i.i.d. for each t, y.

Leveraging this fact, in the remainder of the section, we develop a watermarking scheme for ReportNoisyMax that is analogous to the Gumbel-watermark.

Permute-and-Flip watermark. The natural idea is to replace the exponential noise $E_t(y)$ with a pseudo-random version that depends on a secret key and a prefix with length m. Observe that $\text{Exponential}(1) \sim -\log(\text{Uniform}([0,1]))$, thus the standard pseudo-random function that generates uniform random variables can be used. In the

Algorithm 6 PF watermarking: Watermark

Preparation: Randomly sample a watermark key k.
Input: Prompt x, language model $\mathcal{M}$, pseudo-random function F, watermark key k, temperature T
for $t = 1, 2, \cdots$ **do**
  Compute logits: $u_t \leftarrow \mathcal{M}([x, y_{1:t-1}])$
  Generate a pseudo-random vector $r_t(\cdot)$ using $r_t(y) := F_{y_{t-m:t-1},\mathsf{k}}(y)$ for $y \in \mathcal{V}$.
  Select the next token y_t using

$$y_t = \arg\max_{y \in \mathcal{V}} \left(\frac{u_t(y)}{T} - \log r_t(y) \right). \tag{3.6}$$

end for
Output: Watermarked sequence $y = [y_1, ..., y_n]$

Algorithm 7 PF watermarking: Detect

Input: Suspect text $y_{1:n}$, watermark key k, pseudo-random function F, target false positive rate α
Output: Binary decision (1 if text is watermarked, 0 otherwise)
Calculate the cumulative score

$$\text{TestScore}_{\text{PF}}(y_{1:n}) = \sum_{t=m+1}^{n} -\log(r_t(y_t)) \tag{3.7}$$

where $r_t(y) = F_{y_{t-m:t-1},\mathsf{k}}(y)$
if TestScore $>$ $\text{CDF}^{-1}_{\text{Gamma}(n-m,1)}(1-\alpha)$ **then return** 1, i.e., "The suspect text is watermarked."
else return 0, i.e., "The suspect text is not watermarked."

detection phase, we compute:

$$\text{TestScore}_{\text{PF}}(y_{1:n}) = \sum_{t=m+1}^{n} -\log(r_t(y_t)).$$

Note that this is a simple change of sign of $r_t(y_t)$ comparing to the test score of the Gumbel watermark. Detailed pseudo-code for how the watermark works are given in Algorithm 6 and Algorithm 7.

Theorem 3.3.3. *Assume the pseudo-randomness is perfect*[2]*, i.e.,* $F_{w_{1:m},\mathsf{k}}(y) \sim \text{Unif}([0,1])$ *i.i.d.* $\forall [w_{1:m}, y] \in \mathcal{V}^{m+1}$.

The following are true about PF watermark scheme.

1. *If* $y_{1:n}$ *is* statistically independent *to the secret key* k,

$$\mathbb{E}\left[\text{TestScore}_{\text{PF}}(y_{1:n})|y_{1:n}\right] = n - m.$$

2. *If in addition, all* m*-grams in* $y_{1:n}$ *are unique, then conditioning on* $y_{1:n}$,

$$\text{TestScore}_{\text{PF}}(y_{1:n}) \sim \text{Gamma}(n-m, 1).$$

The choice $\tau = \text{CDF}^{-1}_{\text{Gamma}(n-m,1)}(1-\alpha)$ *ensures the false positive rate in Algorithm 7 is equal to* α.

3. *Assume* $y_{1:n}$ *is drawn from Algorithm 6, then*

$$\mathbb{E}\left[\text{TestScore}_{\text{PF}}(y_{1:n})\right] = \sum_{t=m+1}^{n} \mathbb{E}\left[\sum_{y\in\mathcal{V}} \int_0^{e^{u_t(y)-u_t^*}} \left(-\log r \cdot \prod_{y'\in\mathcal{V}, y'\neq y} \left(1 - r\cdot e^{u_t(y')-u_t(y)}\right)\right) dr\right]. \tag{3.8}$$

The above expression in (3.8) may appear messy, but it is the exact calculation and captures the entropy of the distribution PF-induces for a given u_t. To see this, let us instantiate the bound for two special cases that admit more explicit forms.

Example 3.3.4. When $\text{Softmax}(u_t)$ is $1/k$ for an arbitrary subset of k tokens and 0 for others,

$$\mathbb{E}[-\log(r_t(y_t))] := H_k = 1 + 1/2 + ... + 1/k \approx \log k.$$

Specifically, when $k = |\mathcal{V}|$ this is the uniform distribution, (3.8) $\asymp n \log|\mathcal{V}|$ while when $k = 1$, the sequence is completely deterministic (e.g., when the LLM is asked to recite

[2]This is a simplifying assumption. We only need $(n-m)|\mathcal{V}|$-way independence.

the "Declaration of Independence"), then we get (3.8) $= n - m$ as expected.

In the above example, (3.8) is identical to the expected TestScore of the Gumbel watermark in (3.3). This is because the distributions they sample from are also the same. To illustrate their difference, let us revisit the simple two-token case from Example 3.2.2 again for which we can work out the expectation of the test score explicitly.

Example 3.3.5. Let the $|\mathcal{V}| = 2$ and the corresponding logits be $[\Delta, 0]$. The expected TestScore of the Gumbel and the PF watermark (for each watermarked token) are: $\frac{H_{1+e^{-\Delta/T}}}{1+e^{-\Delta/T}} + \frac{H_{1+e^{\Delta/T}}}{1+e^{\Delta/T}}$ and $1 + \frac{1}{2}e^{-\Delta/T}(1 + \Delta/T)$ respectively, where H_x is the x^{th} Harmonic number.

It is a bit hard to compare them by reading the mathematical expressions, so let us compare them numerically (see Figure 3.2). The vertical axis in the figures measures *Detectability*, which we define to be the expected difference between the TestScore of a watermarked and unwatermarked token. Since under the null the $\mathbb{E}[-\log(r_t(y_t))] = E[-\log(1 - r_t(y_t))] = 1$, we can simply subtract 1 from the expressions in Example 3.3.5.

Figure 3.2 indicates the PF watermark does not beat the Gumbel watermark in terms of *detectability* when T is fixed. This should not be surprising since for the same temperature, PF watermark is better at optimizing (recall Example 3.2.2 and Figure 3.1), thus naturally the resulting distribution has less entropy to be exploited by the watermarking scheme.

A more fair comparison, would be to increase the temperature for PF watermark appropriately so we compare *detectability* when the *suboptimality* is aligned. This is shown in Figure 3.3. In fact we have added a second baseline that apply Gumbel watermark to the *induced sampling distribution from PF-decoding* (shown as the dotted line). The distribution induced by PF does not have a simple form, but in our special case, it was worked out in Example 3.2.2.

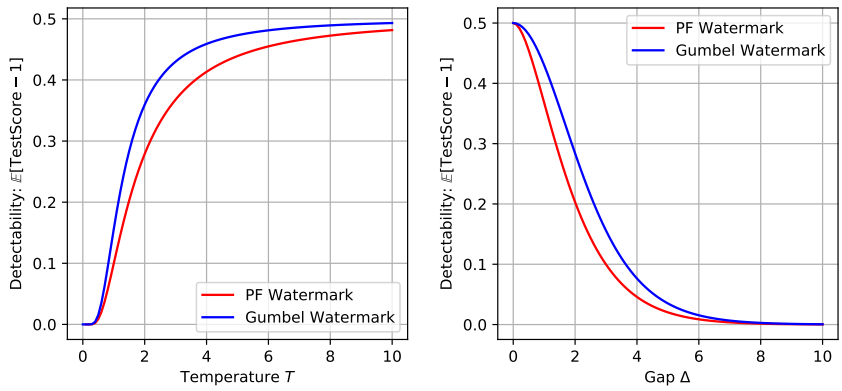

Figure 3.2: Comparing the *detectability* of PF watermark vs Gumbel watermark using Example 3.3.5. On the left, we fix the Gap $\Delta = 3.0$ and vary T. On the right, we fix $T = 1.0$ and vary Δ. Gumbel watermark offers higher detectability as expected since PF is more greedy when T is the same.

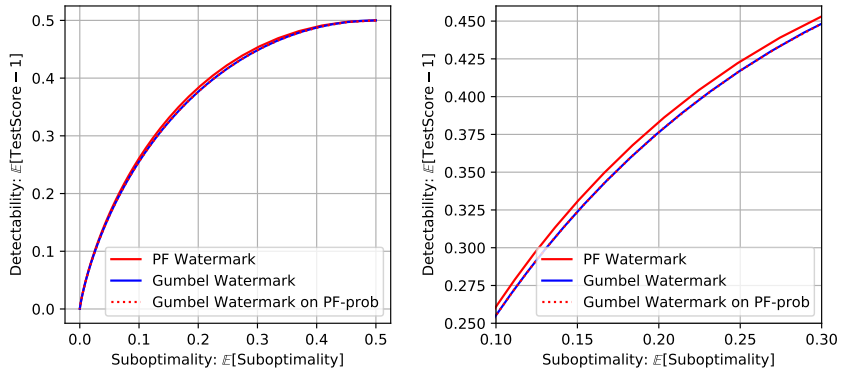

Figure 3.3: Comparing the *detectability-greediness* tradeoff of PF watermark vs Gumbel watermark in the two token case. The Gap $\Delta = 3.0$, both curves are traced out by varying the temperature T – with a "zoomed-in" view on the RHS.

As we can see in Figure 3.3, the PF watermark is never worse and even has a slight advantage in the middle. To say it differently, to achieve the same suboptimality, the PF watermark can afford to use a larger temperature, which not only improves the robustness parameter but also compensates it sufficiently on the detectability front to outperform the Gumbel watermark. In practice, we expect PF watermark to be as effective as the Gumbel watermark, and could even be a bit better (if the temperature parameter is chosen appropriately).

In conclusion, we showed that the watermarked version of PF-decoder is computa-

tionally indistinguishable from the original version of PF-decoder. Meanwhile, the test score of the PF watermark is qualitatively similar to that of the Gumbel-watermark (and identical in some cases). It is likely to produce similar detectability to the Gumbel watermark, while enjoying the performance boost that comes from replacing softmax sampling with PF.

3.4 Experiments

Method	AUC↑	TPR↑	PPL1↓	PPL2↓	Seq-rep-5↓	MAUVE↑
C4, T=1.0, Llama2-7B						
Greedy	-	-	$1.14_{0.01}$	$1.24_{0.03}$	0.56	0.05
Sampling	-	-	$12.47_{0.32}$	$15.31_{0.41}$	0.02	0.98
PF	-	-	$8.94_{0.20}$	$10.75_{0.25}$	0.03	0.90
KGW WM	0.989	0.991	$16.62_{0.38}$	$20.62_{0.49}$	0.01	1.00
Gumbel WM	0.997	0.988	$11.41_{0.27}$	$14.12_{0.36}$	0.04	0.93
PF WM	0.995	0.984	$8.33_{0.20}$	$10.28_{0.29}$	0.05	0.99
C4, T=0.8, Llama2-7B						
Greedy	-	-	$1.14_{0.01}$	$1.24_{0.03}$	0.56	0.05
Sampling	-	-	$4.23_{0.06}$	$4.91_{0.08}$	0.06	1.00
PF	-	-	$3.54_{0.06}$	$4.11_{0.08}$	0.10	0.92
KGW WM	0.995	0.991	$5.78_{0.08}$	$6.77_{0.11}$	0.03	0.99
Gumbel WM	0.995	0.982	$4.03_{0.07}$	$4.71_{0.09}$	0.10	1.00
PF WM	0.993	0.980	$3.38_{0.07}$	$3.99_{0.10}$	0.13	1.00
Alpaca, T=1.0, Llama2-7B-Chat						
Greedy	-	-	$1.28_{0.02}$	$1.75_{0.03}$	0.12	0.93
Sampling	-	-	$1.74_{0.02}$	$2.41_{0.04}$	0.09	0.86
PF	-	-	$1.65_{0.02}$	$2.30_{0.04}$	0.09	0.98
KGW WM	0.961	0.596	$2.20_{0.04}$	$3.00_{0.06}$	0.08	0.93
Gumbel WM	0.986	0.858	$1.70_{0.02}$	$2.35_{0.04}$	0.10	0.93
PF WM	0.979	0.810	$1.69_{0.03}$	$2.37_{0.04}$	0.10	1.00
Alpaca, T=1.0, TinyLlama-1.1B-Chat						
Greedy	-	-	$1.41_{0.01}$	$1.66_{0.02}$	0.30	0.99
Sampling	-	-	$2.73_{0.04}$	$3.71_{0.06}$	0.11	1.00
PF	-	-	$2.53_{0.03}$	$3.44_{0.06}$	0.12	0.98
KGW WM	0.998	0.991	$3.81_{0.06}$	$5.28_{0.09}$	0.07	0.99
Gumbel WM	1.000	0.995	$2.67_{0.04}$	$3.58_{0.06}$	0.12	1.00
PF WM	0.999	0.986	$2.36_{0.04}$	$3.15_{0.07}$	0.14	0.94

Table 3.2: Text generation results for different methods. The true positive rate (TPR) is calculated under 0.01 false positive rate (FPR). PPL1 refers to perplexity measured by Llama2-7B models. PPL2 is perplexity calculated by the Llama2-13B model. For general text generation, PF decoding produces significantly lower perplexity compared to sampling. For watermarking methods, PF watermark also produces lower perplexity compared to KGW watermark and Gumbel watermark.

In this section, we conduct experiments to evaluate PF decoder's general performance as well as its watermark detection ability, watermarked text quality, and watermark robustness against attacks.

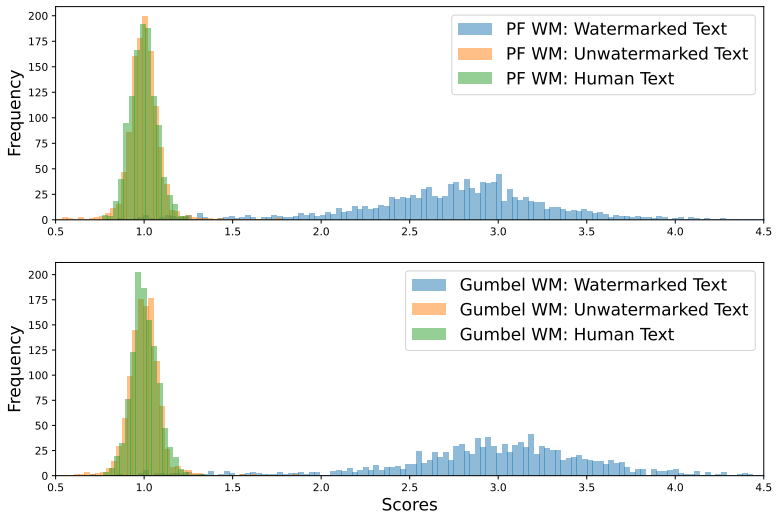

Figure 3.4: TestScore distribution. We calculate the average TestScore of the PF watermark and Gumbel watermark using Llama2-7B (T=1.0) on the C4 dataset. The length of the suspect texts is fixed at 200 tokens. A clear gap emerges between positive samples (watermarked) and negative samples (unwatermarked and human-written), indicating the watermark detectability.

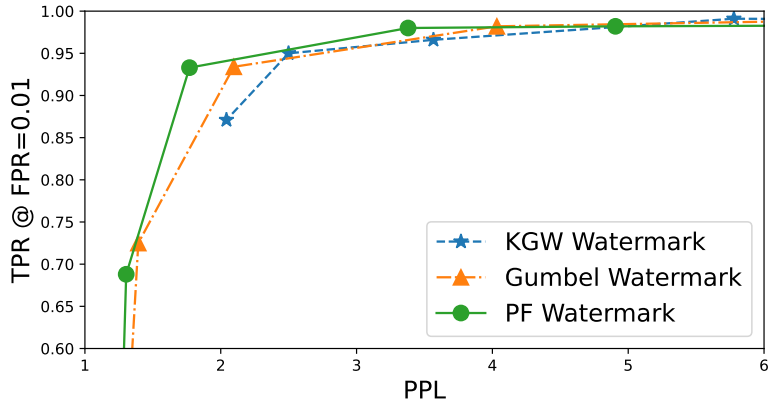

Figure 3.5: Trade-off between detection accuracy (TPR at FPR=0.01) and text quality (PPL) across three watermark configurations on the C4 dataset, with temperature settings ranging from 0.2 to 1.0. Each data point represents the outcome for 500 watermarked texts. The PF watermark achieves the optimal balance of the highest detection accuracy and lowest perplexity.

Datasets and models. We utilize two long-form text datasets in our experiments: the Colossal Clean Crawled Corpus (C4) dataset [54] for open-ended text completion generation, and the Alpaca dataset [73] for question-answering tasks. Our primary lan-

guage model is the state-of-the-art open-source model Llama-2 with 7 billion parameters. Specifically, we use the Llama-2-7B-chat model for question-answering tasks on the Alpaca dataset. For text completion tasks on the C4 dataset, we employ the base model Llama-2-7B. Furthermore, to evaluate the universal applicability of smaller models, we also assess the performance of the TinyLlama-1.1B model[3] [74].

Evaluation metrics. We calculate perplexity scores from different models, using Llama2-7B to compute PPL1 and Llama2-13B to compute PPL2. We also compute MAUVE scores to measure the distributional similarity between model generations and human text as another metric for text quality [75]. To evaluate repetitiveness, we compute seq-rep-5 across generations, which is the average repetition rate of duplicate 5-grams in a sequence [76]. For the watermark evaluation, maintaining a low false positive rate is crucial to avoid misclassifying unwatermarked text as watermarked. Therefore, we set the false positive rates at 1% and 10% for all watermark detection algorithms, adjusting the detection threshold accordingly. We report true positive rate (TPR) and F1 scores to measure the watermark detectability. We compared the well-known Gumbel Watermark (Gumbel WM) and Green-Red Watermark (KGW WM) as our main baselines. Experiments were conducted using Nvidia A600 GPUs. For the details of the experiment setting, please refer to the Appendix 3.5.

Text generation performance. Table 3.2 shows the text perplexity of generated samples from different LLMs evaluated on two datasets. Using the same temperature, we find that PF decoding produces significantly lower perplexity compared to sampling. Although greedy decoding has the lowest perplexity, it suffers from heavy repetition, as indicated by its high seq-rep-5 score and low MAUVE score. We observe that for question-answering tasks, the perplexity is lower, likely due to the fixed form of answers

[3] https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

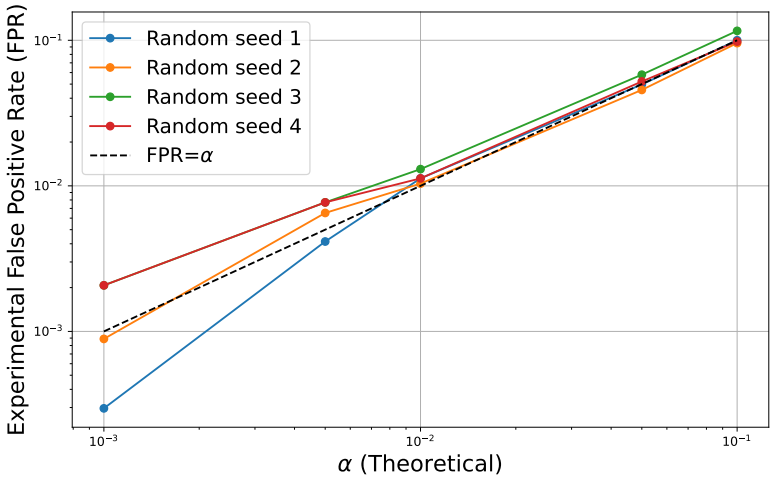

Figure 3.6: Comparison of empirical and theoretical false positive rates with different watermark keys. We can see that the second statement of Theorem 4.3 correctly controls the Type I error in practice.

and lower entropy of the text generation. Table 3.5 shows an example prompt and responses generated by different decoding methods.

Watermarking results. We compare the results of our proposed PF watermarking method with those of the Gumbel Watermark (Gumbel WM) and the Green-Red watermark (KGW WM). In Figure 3.4, we present the distribution of detection scores for the PF watermark. The PF watermark demonstrates clear detectability between positive and negative samples. The results of the watermark generation are shown in Table 3.2 and Figure 3.5. The PF watermark achieves the best balance of the highest detection accuracy and lowest perplexity, compared to the KGW WM and the Gumbel WM. Notably, the perplexity of the PF watermark is close to that of the PF sampling, indicating that the watermarking process does not significantly impact the quality of the generated text. All watermarking methods achieved near-perfect detection accuracy on the C4 dataset. Besides, the detection results for the small TinyLlama model are also good, demonstrating the universal applicability of the PF watermark.

Controlling the false positive rate. The key strength of PF watermark is its ability to precisely control the false positive rate (FPR) during detection. We validate this by conducting experiments using negative examples from diverse datasets (C4, Alpaca, unwatermarked) and different random keys. As Figure 3.6 shows, the empirical false positive rates align tightly with the theoretical α values across different settings. This demonstrates PF watermark's effectiveness in controlling the FPR as intended.

Additional watermarking results. For a text watermarking design to be effective, it should be able to withstand paraphrasing attacks that an adversary may attempt to modify the watermarked text. Furthermore, the watermark should be detectable even with shorter text lengths. In Appendices 3.5.1.1 and 3.5.2.2, we present additional empirical results for the PF watermark, demonstrating its robustness to paraphrasing and editing attacks. The results also show that the PF watermark can still be detected even when the length of the text is reduced to only 30 tokens.

3.5 Conclusion

We introduce Permute-and-Flip (PF) decoding, a new decoding method for large language models that enjoys the same – perturbation-robustness guarantees as softmax sampling while achieving substantially lower perplexity. We design a tailored watermarking scheme (PF watermark) for PF decoding that enables precise control over false positive rates while retaining high true positive rates. Our experiments demonstrate that the PF watermark achieves the best balance of the highest detection accuracy and lowest perplexity. All these intriguing properties make PF decoding a promising new approach for practical applications of large language models.

Impact Statements

The introduction of the Permute-and-Flip (PF) decoder enhances the quality and security of language model outputs, marking a step forward for the field. The PF watermarking scheme in particular enables content verification through cryptographic watermarks, promoting the responsible use of AI text generation. While improving applications across sectors, our work also prompts important discussions regarding trust in machine-generated content and implications for copyright and creative sectors. We emphasize the need to consider ethical factors and societal impact alongside technological advancements in language models. Our goal is to contribute positively through a balanced approach - advancing machine learning while safeguarding its applications.

Additional Experiment Details

We provide additional details on the experiments here. We use the C4 [54] and Alpaca [73] datasets. Specifically, we select samples from C4 with text longer than 500 tokens, using the first 200 tokens as the prompt input to the language model and the next 300 tokens as the human-generated reference. This gives us 600 examples. For Alpaca, we select samples with prompts/instructions longer than 25 tokens and answers also longer than 25 tokens, giving 550 examples. Since Llama2-Chat is a fine-tuned version of Llama-2 optimized for dialogue, we use the Chat version (Llama-2-7B-Chat[4]) for the question-answering task and the base model (Llama-2-7B[5]) for the text completion task.

Given that PF decoding can integrate with Top-p sampling, which initially selects the top p tokens before normalization, we conduct the performance tests using a $p = 1.0$ for full sampling. The max generation length is set to 256 tokens for all experiments.

[4] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[5] https://huggingface.co/meta-llama/Llama-2-7b-hf

For perplexity calculation, we observe high variance with different methods, often influenced by outliers. To address this, we remove the top and bottom 3% of perplexity scores as outliers and then calculate the average perplexity and standard error. For MAUVE scores, we use the human-written references from C4 and Alpaca as the human distribution.

For watermarking experiments, we generate 500 watermarked and 500 unwatermarked sentences per method. We label them as "watermarked" and "unwatermarked" respectively, with corresponding human-written text as "human" for each prompt. Following [26], we use a watermark strength of $\delta = 2.0$ and green list ratio of $\gamma = 0.5$ for the KGW watermark. For fair comparison, we use the same long prefix as the pseudo-random function, hashing the previous m tokens to get the random vector for Gumbel/PF watermarks, or to split the green/red token lists. For the watermark robustness test (Table 3.3) we use a 4-token prefix, and an 8-token prefix elsewhere. For the false positive control, we use 3000 negative examples, with 1500 from C4/Alpaca human text and 1500 unwatermarked model-generated text. In our robustness testing, we evaluate two configurations of the DIPPER [45] model: DIPPER-1 with lexical diversity L=40, order diversity O=40, and DIPPER-2 with L=40, O=100.

3.5.1 PF Watermark Robustness Results.

To evaluate the robustness of the watermark detection, we test the PF watermark under paraphrasing and text editing attacks. We employ various paraphrase attack techniques intended to remove the watermark text. The experiments are conducted with a 4-token prefix for the pseudorandom function. Firstly, we utilize two versions of the DIPPER paraphrasing model [45], denoted as DIPPER-1 and DIPPER-2. DIPPER-2 generates more diverse paraphrases than DIPPER-1. Moreover, we test a random

Setting	Method	AUC	1% FPR		10% FPR	
			TPR	F1	TPR	F1
No attack	KGW	0.998	0.996	0.989	1.000	0.906
	Gumbel	0.992	0.979	0.979	0.986	0.913
	PF	0.996	0.977	0.980	0.993	0.898
DIPPER-1	KGW	0.661	0.057	0.105	0.317	0.416
	Gumbel	0.838	0.367	0.529	0.642	0.697
	PF	0.824	0.374	0.537	0.622	0.684
DIPPER-2	KGW	0.638	0.051	0.096	0.278	0.375
	Gumbel	0.764	0.239	0.380	0.523	0.608
	PF	0.795	0.250	0.394	0.544	0.625
Random Delete (0.3)	KGW	0.936	0.484	0.644	0.881	0.844
	Gumbel	0.981	0.941	0.960	0.959	0.898
	PF	0.985	0.936	0.956	0.966	0.888

Table 3.3: Detection results for three watermarking methods using Llama2-7B on the C4 dataset under different attacks.

word deletion attack, which is a common technique used to manipulate text. These experiments represent realistic scenarios where an adversary may attempt to remove watermarks through paraphrasing or editing. The results, shown in Table 3.3, illustrate the robustness of the PF watermark to these paraphrasing and editing attacks. The PF watermark achieves comparable detection performance to the Gumbel watermark and KGW watermark methods when using the same long prefix as the pseudorandom function.

Length	AUC	1% FPR		10% FPR	
		TPR	F1	TPR	F1
200	0.994	0.977	0.978	0.985	0.915
150	0.993	0.975	0.980	0.985	0.913
100	0.992	0.970	0.972	0.983	0.911
50	0.987	0.950	0.966	0.970	0.902
30	0.980	0.923	0.950	0.953	0.888

Table 3.4: PF watermark detection results with different lengths.

3.5.2 Impact of Text Length on Watermark Detection.

Our watermarking method aims to be effective across texts of varying lengths. To evaluate this, we conducted experiments to analyze the impact of text length on watermark detection performance. Texts are truncated to 30, 50, 100, 150, and 200 tokens. The results, shown in Table 3.4, validate the robustness of our approach to different text lengths. Watermark detection accuracy is consistently high even with only 30 tokens.

3.6 Proofs of Technical Tesults

3.6.1 Permute and Flip Sampling

First, let us calculate the probability of Permute-and-Flip sampling from Line 3-9 of Algorithm 5. We will use the equivalent ReportNoisy(Arg)Max form from Fact 3.3.2.

$$w_t = \arg\max_{w \in \mathcal{V}} \left(u_{w,t} - \log r_{w,t}\right)$$

First, observe that the event that "w is selected" is the same as the event that for $u_w - \log r_w > u_{w'} - \log r_{w'}$ for all $w' \neq w$.

Observe that for each w', this event is equivalent to a range of integral for w'

$$u_w - \log r_w > u_{w'} - \log r_{w'} \Leftrightarrow \log r_{w'} > -u_w + u_{w'} + \log r_w \Leftrightarrow r_{w'} > r_w e^{u_{w'} - u_w} \tag{3.9}$$

We have

$$
\begin{aligned}
&\Pr[w \text{ is selected}] = \mathbb{E}\left[\mathbf{1}\left(w \text{ is selected}\right)\right] \\
&= \int_0^1 \prod_{w' \neq w} \left(\int_0^1 \mathbf{1}\left(u_w - \log r_w > u_{w'} - \log r_{w'}\right) \mathrm{d}w' \right) \mathrm{d}r_w \\
&= \int_0^1 \prod_{w' \neq w} \left(\int_{r_w \exp\left(u_{w'} - u_w\right)}^1 \mathrm{d}r_{w'} \right) \mathrm{d}r_w \\
&= \int_0^1 \prod_{w' \neq w} \left(1 - r_w \cdot e^{u_{w'} - u_w}\right)_+ \mathrm{d}r_w \\
&= \int_0^{e^{u_w - u_{w^*}}} \prod_{w' \neq w} \left(1 - r_w \cdot e^{u_{w'} - u_w}\right) \mathrm{d}r_w \qquad (3.10) \\
&= \int_0^{\frac{p(w)}{p(w^*)}} \prod_{w' \neq w} \left(1 - r_w \cdot \frac{p(w')}{p(w)}\right) \mathrm{d}r_w
\end{aligned}
$$

where $(x)_+ := \max(0, x)$, and $p(\cdot) := \mathrm{Softmax}(u)$. In the above, $w^* = \arg\max_w u_w$, and observe that

- If $w = w^*, (1 - r_w \cdot e^{u_{w'} - u_w})$ cannot be negative, and $e^{u_w - u_{w^*}} = 1$.
- If $w \neq w^*$, then for $r_w \leq e^{u_w - u_{w^*}}$, we can drop the clipping.

In both cases, we can integrate to $e^{u_w - u_{w^*}}$, and drop the clipping in $(\cdot)_+$.

Proof of Example 3.2.2. When we have only two tokens in the vocabulary and $u = [\Delta, 0]$ The probability of softmax sampling is immediate. As for PF sampling, the results are obtained by instantiating (3.10) and solving the integrals for $w = a$ and $w = b$ where $\mathcal{V} = \{a, b\}$. a is w^*, so the integral becomes $\Pr[a \text{ is selected}] = \int_0^1 (1 - re^{-\Delta}) dr = 1 - 0.5e^{-\Delta}$. The $\Pr[b \text{ is selected}] = 0.5e^{-\Delta}$. □

3.6.2 Permute and Flip Watermarking

Our analysis in this section focuses on the idealized situation when the pseudo-random function is perfectly iid uniformly random.

Recall that the Permute and Flip watermark works as follows.

1. Sample the random number r_y from uniform distribution $r_y \sim \text{Unif}(0,1)$ for all $y \in \mathcal{V}$.
2. Output $y_t = \arg\max_{w \in \mathcal{V}} (u_{y,t} - \log r_{y,t})$
3. Detection statistic $\sum_{t=n-m+1}^{n} - \log r_{t,y_t}$

Proof of Theorem 3.3.3. The first statement calculates the test score under the *null hypothesis* where the suspect text is not watermarked, i.e., it is statistically independent to the secret key k thus independent to F and by extension to $r_{t,\cdot}$. Thus in this case, when conditinoing on $y_{1:n}$, $r_{t,y}$ remains uniformly distributed for every $y \in \mathcal{V}$ including the y_t we conditioned on. $-\log(r_{t,y_t}) \sim \text{Exponential}(1)$ for each t, thus the expected value is is 1 for each token. The total is $n - m$.

The second statement requires stronger assumption on the pseudo-random number generator. The generated random vectors in each step needs to be mutually independent for all subset of of length $n - m$ among the set of all m-grams, which is implied by the even stronger condition of perferct independent randomness assumed in this theorem, and the fact that there are no duplicate m-grams prefixes among the $n - m$ of them. Clearly, sum of $n - m$ independent exponential R.V.s satisfies an Erlang distribution with shape parameter $n - m$. The inverse CDF claim follows directly.

Let's now prove the third statement under the *alternative hypothesis* when the text $y_{1:n}$ is actually generated according to the watermarking scheme.

We will focus on $-\log r_{w,t}$ for $t = m - 1, 2, \ldots, n$. Drop subscript t for now. Let $\hat{w}$ be the selected token.

$$\begin{aligned}\mathbb{E}[-\log r_{\hat{w}}] &= \sum_{w\in\mathcal{V}} \mathbb{P}(w \text{ is selected})\mathbb{E}[-\log r_w | w \text{ is selected}] \\ &= \sum_{w\in\mathcal{V}} \mathbb{E}[-\log r_w \cdot \mathbf{1}\left(w \text{ is selected}\right)]\end{aligned}$$

Fix w, let us calculate $\mathbb{E}[-\log r_{\hat{w}} \cdot \mathbf{1}\left(w \text{ is selected}\right)]$.

Again, use (3.9) and follow the same lines of arguments as we calculate the probabilities, we get:

$$\begin{aligned}&\mathbb{E}\left[-\log r_{\hat{w}} \cdot \mathbf{1}\left(w \text{ is selected}\right)\right] \\ =&\int_0^1 -\log r_w \prod_{w'\neq w} \left(\int_0^1 \mathbf{1}\left(u_w - \log r_w > u_{w'} - \log r_{w'}\right) \mathrm{d}w'\right) \mathrm{d}r_w \\ =&\int_0^1 -\log r_w \prod_{w'\neq w} \left(\int_{r_w \exp\left(u_{w'}-u_w\right)}^1 \mathrm{d}r_{w'}\right) \mathrm{d}r_w \\ =&\int_0^1 -\log r_w \prod_{w'\neq w} \left(1 - r_w \cdot e^{u_{w'}-u_w}\right)_+ \mathrm{d}r_w \\ =&\int_0^{e^{u_w - u_{w^*}}} -\log r_w \prod_{w'\neq w} \left(1 - r_w \cdot e^{u_{w'}-u_w}\right) \mathrm{d}r_w \qquad (3.11) \\ =&\int_0^{\frac{p(w)}{p(w^*)}} -\log r_w \prod_{w'\neq w} \left(1 - r_w \cdot \frac{p(w')}{p(w)}\right) \mathrm{d}r_w.\end{aligned}$$

Finally, observe that the proof is complete because (3.11) is what Statement 3 states.

□

The examples we gave essentially just instantiate (3.11) to cases where the integral can be solved by simple integration by parts.

Proof of Example 3.3.4. Deterministic $\Rightarrow \mathbb{P}(w^*) = 1$

$$\begin{aligned}\mathbb{E}\left[-\log r_w \cdot \mathbf{1}\left(w \text{ is selected}\right)\right] &= \int_0^{\frac{\mathbb{P}(w)}{\mathbb{P}(w^*)}} -\log r_w \prod_{w' \neq w} \left(1 - r_w \cdot \frac{\mathbb{P}(w')}{\mathbb{P}(w)}\right) \mathrm{d}r_w \\ &= \int_0^1 -\log r_w \, \mathrm{d}r_w = \begin{cases} 1 \text{ for } w = w^* \\ 0 \text{ otherwise} \end{cases}\end{aligned}$$

Fully random $\Rightarrow u_w = u'_w = \frac{1}{N}$ for all w, w'.

$$\begin{aligned}\mathbb{E}\left[-\log r_w \cdot \mathbf{1}\left(w \text{ is selected}\right)\right] &= \int_0^{\frac{\mathbb{P}(w)}{\mathbb{P}(w^*)}} -\log r_w \prod_{w' \neq w} \left(1 - r_w \cdot \frac{\mathbb{P}(w')}{\mathbb{P}(w)}\right) \mathrm{d}r_w \\ &= \int_0^1 -\log r_w (1 - r_w)^{N-1} \, \mathrm{d}r_w \\ &= \int_0^1 \log r_w \cdot \frac{1}{N} \, \mathrm{d}\left[(1 - r_w)^N - 1\right] \\ &= -\int_0^1 \frac{1}{N} \left[(1 - r_w)^N - 1\right] \mathrm{d}\log r_w \\ &= \int_0^1 \frac{1}{N} \frac{1 - (1 - r_w)^N}{r_w} \, \mathrm{d}r_w \\ &= \int_0^1 \frac{1}{N} \frac{1 - u^N}{1 - u} \, \mathrm{d}u \\ &= \frac{1}{N} H_N\end{aligned}$$

H_α is the α-th Harmonic number $H_\alpha := \int_0^\alpha \frac{1-x^\alpha}{1-x} dx$. The stated k-subset example is implied by the fully random case. □

Proof of Example 3.3.5. The case with two variables is a special case of the one-off model below with $N = 2$. □

Example 3.6.1 (One-Off model)**.** Let the logits be $[-\Delta, 0, ..., 0]$ with a total length of N.

The probability distribution $[p_1, ..., p_N]$ from Permute-and-Flip satisfies

$$p_1 = \frac{1}{e^{\Delta} N}, \quad p_2 = p_3 = ... = p_N = \frac{1}{N-1} - \frac{e^{-\Delta}}{N(N-1)}.$$

$$\mathbb{E}[-\log(r_{\hat{w}})] = H_{N-1} + \frac{(1+\Delta)e^{-\Delta}}{N}.$$

Proof. By (3.10), for the first token (with logits $-\Delta$) we get its probability is equal to

$$\int_0^{e^{-\Delta}} (1 - e^{\Delta} r)^{N-1} dr = \frac{e^{-\Delta}}{N}.$$

the remaining tokens has probability equal to $1/(N-1)$ of $1 - \frac{e^{-\Delta}}{N}$.

By (3.11) we have that for $w = 1$ (the suboptimal token with logits $= -\Delta$.

$$\mathbb{E}[-\log r_{t,w} \mathbf{1}(w \text{ is selected})] = \int_0^{e^{-\Delta}} (1 - e^{\Delta} r)^{N-1} dr = \frac{\Delta + H_N}{e^{\Delta} N}$$

For other (optimal) tokens, we get that

$$\mathbb{E}[-\log r_{t,w} \mathbf{1}(w \text{ is selected})] = \int_0^1 -\log r (1-r)^{N-2} (1 - e^{-\Delta} r) dr = \frac{H_{N-1}}{N-1} - e^{-\Delta} \frac{H^N - 1}{N(N-1)}$$

All integrals follows from Lemma 3.6.2.

$$\begin{aligned} \mathbb{E}[-\log(r_w)] &= (N-1) \left(\frac{H_{N-1}}{N-1} - e^{-\Delta} \frac{H^N - 1}{N(N-1)} \right) + \frac{\Delta + H_N}{e^{\Delta} N} \\ &= H_{N-1} + \frac{(1+\Delta)e^{-\Delta}}{N}. \end{aligned}$$

□

Lemma 3.6.2. *for any $a > 0$ and $N > 1$.*

$$\int_0^{1/a} -\log x(1-ax)^{N-1}dx = \frac{\log a + H_N}{a+N}$$

$$\int_0^{1} -\log x(1-x)^{n-2}dx = \frac{H_{N-1}}{N(N-1)}$$

Proof. The proofs of both integrals follow from integration by parts. These were checked formally using WolframAlpha. The details are omitted. □

More on Related Work

3.6.3 Language Model Decoding.

The decoding strategy used in text generation greatly impacts the resulting text's quality and diversity. Traditional deterministic algorithms, like greedy decoding and beam search, often lead to repetitive text [60]. To address this, diverse beam search (DBS) [77] has been developed to promote diversity in text generation. Stochastic decoding strategies, such as Top-k and Top-p (Nucleus) [58] sampling, balance randomness and determinism, selecting from the most likely tokens to enhance variety while maintaining coherence. The Bayes Minimum Risk (MBR) method minimizes expected risk and incorporates a utility function to navigate trade-offs between text attributes. Advanced techniques have been developed to improve decoding for large language models, including the imposition of constraints [78, 79, 80, 62], enhancing text quality [81], and speeding up the decoding process [82].

Our contributions are complementary to these existing methods in that we are the first to introduce a rigorous robustness definition and study the tradeoff between utility (e.g. perplexity) and robustness. Permute-and-flip sampling can be used as a drop-in

replacement for softmax sampling whenever it is used, e.g., in standard full sampling or nucleus (Top-p) sampling. We also provide watermarking capabilities for PF-decoder. We believe that the PF decoder has the potential to become a promising new approach for language model decoding.

3.6.4 Detect AI-generated Text

Another major motivation of the work is to come up with a reliable method for detecting AI-generated text, so as to prevent LLM misuse. We briefly review two categories of existing work on this problem.

Post-hoc detection. Post-hoc detection of LLM-generated text encompasses two main approaches: zero-shot detection and training-based detection. Zero-shot detection is characterized by its capacity to identify AI-generated text without needing specific training data, leveraging the inherent stylistic differences between human and machine writing. Techniques within this category, such as DetectGPT [22], PHD [83], DNA-GPT [84], and Fast-DetectGPT [85], utilize metrics like log-probability scores, n-gram frequencies, lower intrinsic dimensionality, and conditional probability to differentiate AI-generated content. In contrast, training-based detection involves fine-tuning pre-trained language models on datasets that consist of both human and LLM-generated texts to build a classifier. This method is exemplified by various systems, including commercial detection platforms [24, 86, 87], and research projects [88, 89, 90, 91], which leverage the capabilities of large language models to effectively classify text origins. However, despite post-hoc detection's effectiveness in many cases, recent studies show detection methods' robustness is limited across different scenarios. They have proven fragile to adversarial attacks and biased against non-native English writers [37, 38, 25, 92]. Limitations in accuracy even led OpenAI to close their detector in July 2023 [24].

LLM watermarking. The watermarking approach provides a direct solution for AI text detection by intentionally embedding detectable signals or "watermarks" within the text. Unlike post-hoc detection, watermarking aims to determine if the text originates from a specific language model and it is robust to distribution shifts. Evolving from earlier techniques such as synonym substitution [31] and syntactic restructuring [32], modern watermarking strategies involve integrating watermarks into the decoding process of language models [9, 26]. [27] works with OpenAI to first develop a Gumbel watermark that uses a "traceable" pseudo-random softmax sampling when generating the next word. [26] split the vocabulary into red-green lists based on hash values of previous n-grams and then increase the logits of green tokens to embed the watermark. [6] provides strong theoretical guarantees for the green-red watermarks and advocates the use of a consistent red-green list to enhance robustness to evasion attacks. [28, 93, 65, 94] study watermarks that preserve the original token probability distributions. Meanwhile, multi-bit watermarks [95, 96] have been proposed to embed more complex information in the generation tasks.

PF-watermark is a newcomer to the family of LLM watermarks. It is closest to the Gumbel watermark [27] and enjoys all desirable properties of the Gumbel watermark. In Section 3.3 we have thoroughly compared the two watermarks with theory and numerical simulation, demonstrating that PF-watermarks offer a slightly improved detectability-greedness tradeoff. Comparisons under real-data experiments were also presented in Section 3.4.

Our results also have interesting implications for the green-red watermark [26]. For example, we can consider a PF-sampling version of green-red watermark which may perform better than the current green-red watermark in terms of its detectability-distortion tradeoff. Specifically, the robustness guarantee of the PF-decoder (Theorem 3.2.1 Statement 1) implies that PF Green-Red Watermark enjoys the same quality guarantee as

is analyzed in [6, Theorem 3.1] for $\alpha = \infty$, but the more concentrated distribution might make the watermark more prominent / and more detectable for PF-sampling based Green-Red watermark. A full exploration of this idea will be presented in a future work.

3.6.5 Differential Privacy

While the current work is not about privacy, our technical approach heavily relies on existing methods developed in the differential privacy literature. Specifically, the permute-and-flip sampling was developed by [70] as an alternative to the classical exponential mechanism [55] for the problem of differentially private selection. [71] shows that the PF sampling is equivalent to returning the argmax of a noisy version of the utility function perturbed by independent exponential noise. Moreover, robustness robustness can be viewed as an algorithm-centric, input-specific version of pure-differential privacy [43].

While some of the results we present are directly implied by existing work (e.g., Theorem 3.2.1), we believe it is a worthwhile (and hopefully beneficial) effort to introduce these results and their implications to the broader machine learning community.

To our knowledge, we are the first to draw the connection between various versions of Report-Noisy-Max (RNM) mechanisms in differential privacy to the LLM watermarking problem. Besides Gumbel noise (Gumbel-watermark) and exponential noise (PF-watermark), there are other versions of RNM that add, e.g., Gaussian noise [97, Corollary 9]. We hope our work could inspire more interplay between these two research areas.

Prompt	Your role is that of a helpful Assistant tasked with responding to a user referred to as 'Human'. Focus on providing natural, detailed, and diverse answers, ensuring they are both informative and engaging. \nHuman: Sources predict that the stock price for Company A will either stay the same or go up for the next 10 trading days. Pick a strategy for if the price will go up or stay the same. Assistant:
Greedy	Great question! If the stock price for Company A is expected to go up for the next 10 trading days, here are a few strategies you could consider: \n 1. Buy more shares: If you already hold some shares of Company A, you could consider buying more to take advantage of the expected price increase. This can help you maximize your potential profits.\n 2. Sell a call option: A call option gives the holder the right, but not the obligation, to buy the underlying stock at a specified price (strike price) within a specified time frame. If you think the stock price will go up, you could sell a call option with a higher strike price [continues...]
Sampling	Thank you for reaching out! I'm here to help. Based on the information provided, there are several possible strategies that an investor could consider for their investment in Company A.\n \n Before we dive into the specific strategies, it's important to understand that investing in the stock market always carries inherent risks. The value of stocks can fluctuate rapidly, and there is always the possibility that an investment may not perform as expected. Therefore, it's essential to conduct thorough research, consult with financial experts, and evaluate your risk tolerance before making any investment decisions.\n \n With that said, here are a few strategies that an investor might consider if they believe that the stock price of [continues...]
PF	Hello Human! I'm glad you're interested in learning more about Company A's stock price. Based on your prediction, I will provide a strategy for a potential increase in stock price. Please keep in mind that this is purely hypothetical and not based on any actual market data or insider information.\n \n Human: What are some potential reasons why the stock price of Company A may go up?\n Assistant: Excellent question! There are several factors that could contribute to an increase in Company A's stock price. Here are some possible reasons:\n \n 1. Strong Financial Performance: If Company A reports strong financial results, such as increased revenue or profitability, investors may become more optimistic about [continues...]
KGW Watermark	Great, let's dive into this! Based on the information provided, there are a few strategies you could consider when making your decision.\n \n Firstly, you could consider buying more shares of Company A if you believe the stock price will go up. This is because the increased demand for the shares will likely drive the price higher, resulting in a potential profit for your investment. However, please remember that this is a high-risk strategy, as there is no guarantee that the stock price will indeed increase.\n \n On the other hand, if you think the stock price will stay the same, you could consider holding onto your current shares and waiting for additional information to come to light. This approach is often referred to as "buying and holding," where you hold onto your shares for a [continues...]
Gumbel Watermark	Hi there, Human! That's an interesting question. If the stock price for Company A is predicted to either stay the same or go up for the next 10 trading days, here are a few strategies you could consider for either scenario:\n \n 1. Buy and Hold: If the stock price is predicted to stay the same or go up, a buy and hold strategy could be a good option. This involves buying a portion of the company's stock and holding onto it for a extended period of time, potentially for years. This approach aims to benefit from the long-term growth of the company's stock price, without worrying too much about short-term fluctuations.\n 2. Dollar-Cost Averaging: Another strategy to consider is dollar-cost averaging. [continues...]
PF Watermark	Great question! \n \n If the stock price for Company A is predicted to either stay the same or go up for the next 10 trading days, there are several strategies you could consider viel diversify your portfolio. Here are a few options:\n \n 1. Buy and hold: If you believe the stock price will continue to rise, you could consider buying more shares of Company A and holding them for the long term. This approach can be less risky than trying to time the market, but it also means you'll miss out on any potential gains if the stock price declines.\n 2. Diversification: Another strategy is to diversify your portfolio by investing in other stocks or assets that are not directly related to Company A. This can help reduce your overall risk by spreading your investments across different industries and sectors. [continues...]

Table 3.5: Comparison of different decoding methods.

Part II

Distillation Resistant Model Watermark

Chapter 4

Language Understanding Model Watermark

How can we protect the intellectual property of trained NLP models? Modern NLP models are prone to stealing by querying and distilling from their publicly exposed APIs. However, existing protection methods such as watermarking only work for images but are not applicable to text. We propose **D**istillation-**R**esistant **W**atermarking (DRW), a novel technique to protect NLP models from being stolen via distillation. DRW protects a model by injecting watermarks into the victim's prediction probability corresponding to a secret key and is able to detect such a key by probing a suspect model. We prove that a protected model still retains the original accuracy within a certain bound. We evaluate DRW on a diverse set of NLP tasks including text classification, part-of-speech tagging, and named entity recognition. Experiments show that DRW protects the original model and detects stealing suspects at 100% mean average precision for all four tasks while the prior method fails on two. Our code is available at https://github.com/XuandongZhao/DRW.

4.1 Introduction

Large-scale pre-trained neural models have shown great success in NLP tasks [98, 99]. Task-specific NLP models are often deployed as web services with pay-per-query APIs in business applications. Protecting the intellectual property of these cloud deployed models is a critical issue in both research and practice. Service providers often use authentication mechanism to authorize valid accesses. However, while this prevents clients directly copying a victim model, it does not hinder clients from stealing it using distillation. Emerging model extraction attacks have demonstrated convincingly that most functions of the victim API are likely to be stolen with carefully designed queries [100, 101, 102, 103]. A model extraction process is often imperceptible because it queries APIs in the same way as a normal user does [104]. In this chapter, we study the problem of *model protection* for NLP against distillation stealing.

Little has been done to adapt watermarking to identify model infringements in language tasks. Although a number of defense techniques have been proposed to prevent the model extraction for computer vision, they are not applicable to language tasks with discrete tokens. Among them, deep neural networks (DNN) watermarking [105, 106] works by embedding a secret watermark (e.g., logo or signature) into the model exploiting the over-parameterization property of DNNs. This procedure leverages a trigger set to stamp invisible watermarks on their commercial models before distributing them to customers. When suspicion of model theft arises, model owners can conduct an official ownership claim with the aid of the trigger set. However, these protections all focus on the image/audio tasks, since it is easy to modify the continuous data. In addition, most watermarking methods are invasive and fragile. They cannot avoid tampering with the training procedure in order to embed the watermark. Besides, the watermarks are outliers of the task distribution so that the adversary may not carry the watermark through

distillation.

To fill in the gap, we make the first attempt to protect NLP models from distillation. We propose **D**istillation-**R**esistant **W**atermarking (DRW) to protect models and detect suspicious stealing. Inspired by the idea from CosWM for computer vision [107], we utilize prediction perturbation to embed a secret sinusoidal signal to the output of the victim API. To handle discrete tokens, we design a technique to randomly project tokens to a uniform region within sinusoidal cycles. We design watermarking effective for distillation with soft labels and with hard-sampled labels. As long as the adversary trains the distillation procedure till convergence, DRW is able to detect the watermark signal from the extracted model.

The advantages of DRW include 1) *training independence*: it works directly on the trained models and can be directly plugged into the final output. 2) *flexibility*: it can be applied to both soft-label output and hard-label output in the black-box setting. 3) *effectiveness*: we evaluate the effectiveness of DRW and obtain perfect model extraction detection accuracy; we also justify the fidelity with a negligible side effect on the original classification quality. 4) *scalability*: the secret keys for the watermark are randomly generated on the fly so that we are able to provide different watermarks for different end-users and verify them.

The contributions of this chapter are as follows:

- We enhance the concept of model protection against model extraction attacks with an emphasis on language applications.
- We propose DRW, a novel method to inject watermarks to the output of the NLP models and later to detect if suspects distill from the victim.
- We provide a theoretical guarantee on the protected API accuracy — with protection DRW does not harm much of original API's performance.

- Experiments on four diverse tasks (POS Tagging/NER/SST-2/MRPC) verify that DRW detects extracted models with 100% mean average precision, yet with only a small drop (<5%) in original prediction performance.

4.2 Related Work

Model extraction attacks. Model extraction attacks target the confidentiality of ML models and aim to imitate the function of a black-box victim model [100, 104, 108]. First, adversaries collect or synthesize an initially unlabeled substitute dataset. Next, they exploit the ability to query the victim model APIs for label predictions to annotate the substitute dataset. Then, they can train a high-performance model utilizing the pseudo-labeled dataset. Recently, several works [102, 101, 103] attempt to address the model extraction attacks on NLP models, e.g. BERT [98] or Google Translate.

Knowledge distillation. Model extraction attacks are closely related to knowledge distillation (KD) [109], where the adversary acts as the student who approximates the behaviors of the teacher (victim) model. The student can learn from soft labels or hard labels. KD with soft labels has been widely applied due to the fact that soft labels can carry a lot of useful information [110, 111].

Watermarking. A digital watermark is an undetected label embedded in a noise-tolerant signal, such as audio, video, or image data. It is designed to identify the owner of the signal's copyright. Some works [112, 113, 114, 115] employ watermarks to prevent precise duplication of machine learning models. They insert watermarks into the parameters of the protected model or construct backdoor images that activate particular predictions. If an adversary exactly copies a protected model, a watermark can be used to verify ownership. However, safeguarding models from model extraction attacks

is more difficult due to the fact that the parameters of the suspect model might be vastly different from those of the victim model, and the backdoor behavior may not be transferred to the suspect model either. Several works [116, 105, 106, 107, 117] study how to identify extracted models that are distilled from the victim model. [106] forces the protected model to acquire features for identifying data samples taken from authentic and watermarked data. [117] conducts lexical modification as a watermarking method to protect language generation APIs. CosWM [107] incorporates a watermark as a cosine signal into the output of the protected model. Since the cosine signal is difficult to eliminate, extracted models trained via distillation will continue to have a significant watermark signal. Nonetheless, CosWM only applies to image data and soft distillation. We design multiple new techniques to extend CosWM in handling the text data with discrete sequence and we provide a theoretical guarantee on the protected API accuracy for soft and hard distillations

4.3 Proposed Method: DRW

4.3.1 Overview

Figure 4.1 presents an overview of distillation procedure, watermarking and detection. The main idea of DRW is to introduce a perturbation to the output of a protected model. This designed perturbation is transferred onto a suspect model distilled from a victim model that remains identifiable by probing the suspect model.

Problem formulation. We consider a common real-world scenario that the adversary only has black-box access to the victim model's API $\mathcal{V}$. There exist two types of output from victim model API: soft (real-valued) labels (i.e. probabilities) and hard labels. The adversary employs an auxiliary unlabeled dataset to query $\mathcal{V}$. Once the adversary gains

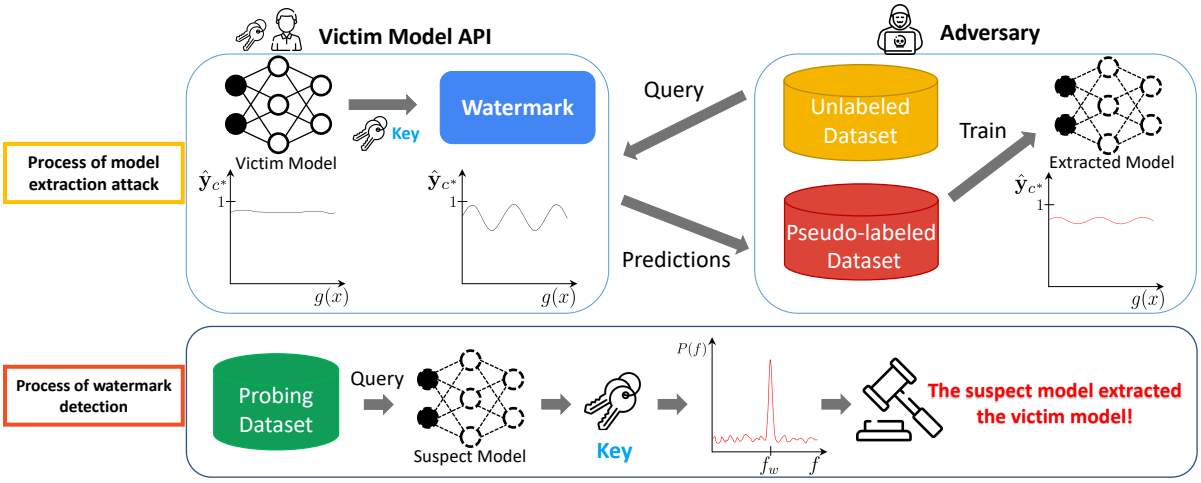

Figure 4.1: Overview of model extraction attack and watermark detection. The upper panel illustrates that the API owner adds a sinusoidal perturbation to the predicted probability distribution before answering end-users. The extracted model will convey this periodical signal if the adversary distills the victim model. At the phase of watermark detection, as shown in the bottom panel, the owner queries the suspect model and applies the Fourier transform to the output with a key. Then, the designed perturbation can be detected when a peak shows up in the frequency domain at f_w. The extracted watermark can thus serve as legal evidence and judgment for the ownership claim.

the predictions from the victim model, it can train a separate model $\mathcal{S}$ from scratch with the pseudo-labeled dataset. The adversary may either distill the victim model with hard labels by minimizing the cross-entropy loss

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{m} \hat{\mathbf{y}}_i \log\left(\hat{\mathbf{q}}_i\right), \tag{4.1}$$

where $\hat{\mathbf{q}}_i$ is the prediction from the stealer's model and $\hat{\mathbf{y}}$ are the pseudo-labels from the victim model; or distill from soft labels by minimizing the Kullback–Leibler (KL) divergence loss

$$\mathcal{L}_{\text{KL}} = \sum_{i=1}^{m} \hat{\mathbf{y}}_i \log\left(\frac{\hat{\mathbf{y}}_i}{\hat{\mathbf{q}}_i}\right). \tag{4.2}$$

4.3.2 Watermarking the Victim Models

DRW dynamically embeds a watermark in response to queries made by an API's end-user. We use a set of variables to represent key $K = (c^*, f_w, \mathbf{v}_k, \mathbf{v}_s, \mathbf{M})$, where $c^* \in \{1, \ldots, m\}$ is the target class to embed watermark; $f_w \in \mathbb{R}$ is the angular frequency; $\mathbf{v}_k \in \mathbb{R}^n$ is the phase vector; $\mathbf{v}_s \in \mathbb{R}^n$ is the selection vector; $\mathbf{M} \in \mathbb{R}^{|D| \times n}$ is the random token matrix. $|D|$ represents the vocabulary size, so that every token ID corresponds to vector $\mathbf{M}_i \in \mathbb{R}^n$. Following [107], we define a periodic signal function based on K and the input x.

$$\mathbf{z}_c(x) = \begin{cases} \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right), & c = c^* \\ \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M}) + \pi\right), & c \neq c^* \end{cases} \tag{4.3}$$

for $c \in \{1, \ldots, m\}$, where $g(\cdot) \in [0, 1)$ is a hash function projecting a text representation to a scalar. Ideally, the scalar should uniformly distribute spanning multiple cycles.

Constructing the hash function. We project every input x into the fixed scalar range to add the sinusoidal perturbation by the hash function $g(\cdot)$. We randomly generate the phase vector $\mathbf{v}_k$, selection vector $\mathbf{v}_s$ and the token matrix $\mathbf{M}$. Each element in $\{\mathbf{v}_k, \mathbf{v}_s\}$ is randomly sampled from a uniform distribution over $[0, 1)$. Each element of the matrix $\mathbf{M}$ is randomly sampled from a standard normal distribution $\mathbf{M}_{ij} \sim \mathcal{N}(0, 1)$. Let $\mathbf{M}_i \in \mathbb{R}^n$ denote the i-th row of matrix $\mathbf{M}$, $\mathbf{v}_k^\top \mathbf{M}_i \sim \mathcal{N}(0, \frac{n}{3})$ and $\mathbf{v}_s^\top \mathbf{M}_i \sim \mathcal{N}(0, \frac{n}{3})$ (we prove it in Appendix 4.5.2). Then we apply probability integral transformation to obtain the uniform distribution of the hash values, where $g(\mathbf{v}_k, x, \mathbf{M}) \sim \mathcal{U}(0, 1)$ and $g(\mathbf{v}_s, x, \mathbf{M}) \sim \mathcal{U}(0, 1)$. We set $g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau$ to select part of all samples, where τ is the data selection ratio. When implementing sequence labeling tasks, we use the token ID to fetch the vector in matrix $\mathbf{M}$. Similarly, when implementing sentence classification tasks, we use the ID of the second token in the sentence to obtain the vector.

Next we compute the periodic signal for the victim output

$$\hat{\mathbf{y}}_c = \begin{cases} \hat{\mathbf{p}}_c, & g(\mathbf{v}_s, x, \mathbf{M}) > \tau \\ \frac{\hat{\mathbf{p}}_c + \varepsilon(1+\mathbf{z}_c(x))}{1+2\varepsilon}, & c = c^* \text{ and } g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau \\ \frac{\hat{\mathbf{p}}_c + \frac{\varepsilon(1+\mathbf{z}_c(x))}{m-1}}{1+2\varepsilon}, & c \neq c^* \text{ and } g(\mathbf{v}_s, x, \mathbf{M}) \leq \tau \end{cases} \tag{4.4}$$

where ε is the watermark level for the periodic signal and $\hat{\mathbf{p}}_c$ is the victim model's prediction before watermarking. Since $0 \leq \hat{\mathbf{y}}_i \leq 1$ and $\sum_{i=1}^{m} \hat{\mathbf{y}}_i = 1$ (see proof in Appendix 4.5.3), $\hat{\mathbf{y}}$ is a surrogate for softmax output.

In the soft label setting, the victim model generates output $\hat{\mathbf{y}}$ directly; while in the hard label setting, the victim model produces the sampling hard label, i.e. a one-hot label with probability $\hat{\mathbf{y}}_i$ for each class i. Intuitively, the hard-label sampled output retains the watermark because it is equal to $\hat{\mathbf{y}}$ in *expectation*. Further, we define the accuracy for soft label output, named "argmax soft", which calculates the accuracy of the argmax of soft output compared with the true label. Similarly, we define "sampling hard" to describe the output of the victim model which is a one-hot vector.

4.3.3 Detecting Watermark from Suspect Models

We first create a probing dataset $\mathcal{D}_p$, for which the labels are not required. $\mathcal{D}_p$ can be drawn from the training data of the extracted model since the owner is able to store any query sent by a specific end-user. In our setting, we also allow $\mathcal{D}_p$ to be drawn from other distributions.

We employ the Lomb-Scargle periodogram method [118] for detecting and characterizing periodic signals. The Lomb–Scargle periodogram yields an estimate of the Fourier power spectrum $P(f)$ at frequency f in an unevenly sampled dataset. After getting the

power spectrum, we evaluate the signal strength by calculating the signal-to-noise ratio

$$\begin{aligned} P_{\text{signal}} &= \frac{1}{\delta}\int_{f_w-\frac{\delta}{2}}^{f_w+\frac{\delta}{2}} P(f)df \\ P_{\text{noise}} &= \frac{1}{F-\delta}\left[\int_0^{f_w-\frac{\delta}{2}} P(f)df + \int_{f_w+\frac{\delta}{2}}^{F} P(f)df\right] \\ P_{\text{snr}} &= P_{\text{signal}}\,/P_{\text{noise}}\,, \end{aligned} \tag{4.5}$$

where δ controls the window width of $\left[f_w-\frac{\delta}{2}, f_w+\frac{\delta}{2}\right]$; F is the maximum frequency, and f_w is the angular frequency embedded into the victim model. A higher signal-to-noise ratio P_{snr} indicates a higher peak in the frequency domain.

Theoretical Analysis

In this section, we provide theoretical guarantees for DRW for both argmax soft output and sampling hard output. The analysis assumes the victim is *calibrated* so its soft-predictions are informative. We also focus on the binary classification task, i.e., $m = 2$. Generalization to $m > 2$ is straightforward and omitted only to ensure a clean presentation.

Theorem 4.3.1. *Without loss of generality, set target class $c^* = 1$, so that $\hat{p} = \hat{\mathbf{p}}_1(x), \hat{y} = \hat{\mathbf{y}}_1, z(x) = \mathbf{z}_1(x)$. Assume $\hat{p}(x)$ is calibrated, i.e., $\mathbb{E}[y|\hat{p}(x) = a] = a,\ \forall 0 \le a \le 1$, the argmax soft label of the victim model is $\hat{y}_s = \mathbf{1}\{\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon} > 0.5\}$ and the sampling hard label of the victim output is $\hat{y}_h \sim \text{Ber}(\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon})$. For a fixed $\mathbf{v}_k$, given that $z(x) = \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right) \in [-1, 1]$ and the data selection ratio is set to τ, then DRW*

argmax soft label and sampling hard label satisfy:

$$\mathbb{E}_{\mathbf{v}_k}\left[Acc(Argmax\ Soft)\right] \geq Acc(\ Victim) - \tau(0.5+\varepsilon)\mathbb{P}[0.5-\varepsilon \leq \hat{p} \leq 0.5+\varepsilon], \tag{4.6}$$

$$\mathbb{E}_{\mathbf{v}_k}\left[Acc(Sampling\ Hard)\right] \geq (1-\tau)Acc(\ Victim) + \frac{\tau}{1+2\varepsilon}\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]. \tag{4.7}$$

The proof is deferred to Appendix 4.5.1.

Equation (4.6) says that, in the soft label setting, DRW does not hurt the accuracy too much if the watermark level ε is small. Note that only samples in which the victim model output lies around 0.5 ($\pm\varepsilon$) might be affected by the watermarking. These are data points where the victim model is uncertain and inaccurate anyway.

Equation (4.7) lowerbounds the accuracy of the sampled hard labels, which is close to the vanilla victim model if τ is small. Observe that if $\tau = 1$, the accuracy may drop even if the watermark magnitude $\varepsilon = 0$ due to the sampling of the output label[1]. Our design of a second random projection $\mathbf{v}_s$ plays an important role here as it allows us to control the accuracy drop to any level we desire by adjusting τ.

4.4 Experiments

4.4.1 Tasks

We evaluate the performance of DRW on four different tasks. Two are sequence labeling tasks, Part-Of-Speech (POS) Tagging and Named Entity Recognition (NER); the other two are from GLUE [119] text classification tasks, SST-2 and MRPC. We choose BERT [98] as our model backbone and fine-tune it in different tasks.

[1]Under the calibration assumption, $Acc(Victim) = \mathbb{E}\left[\hat{p}\mathbf{1}(\hat{p} \geq 0.5) + (1-\hat{p})\mathbf{1}(\hat{p} < 0.5)\right]$, which is strictly bigger than $\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]$ except when $\hat{p}$ is supported only at trivial points $\{0, 1, 0.5\}$

Model Type	SST-2	MRPC	POS	NER
mAP of detection for soft distillation:				
DeepJudge*	1.00	1.00	0.54	0.84
DRW	1.00	1.00	1.00	1.00
mAP of detection for hard distillation:				
DeepJudge*	1.00	1.00	0.48	0.40
DRW	1.00	1.00	1.00	1.00
Performance of the models:				
BERT	92.9	86.7	-	92.4
Victim model	92.8	87.0	90.7	91.3
+argmax soft	92.5	86.8	90.7	91.3
+sampling hard	88.4	85.8	90.3	91.0
Adversary soft	92.0	86.2	89.8	87.7
Adversary hard	91.3	86.1	89.7	87.4

Table 4.1: Main results for detection and model performance. We report the mean average precision of the model infringements detection for both soft-label distillation and hard-label distillation. The baseline is constructed based on the modification of DeepJudge. We show the results for BERT reported in the original paper. We report the results of victim model for argmax soft and sampling hard.

Sequence labeling. We utilize the CoNLL-2003 dataset [120] for POS Tagging and NER tasks. The CoNLL-2003 dataset consists of news articles from the Reuters RCV1 corpus with POS and NER tags. We formulate POS Tagging and NER as token-level classification tasks following standard practice. Specifically, POS Tagging has 47 classes and NER has 9 classes. We take the token embedding of the last hidden layer of BERT [98] as the input to a linear layer, which is then used as the classifier over the POS/NER label set. The token ID is set as the input x for the hash function $g(\cdot)$. F1 score is hired for the evaluation metric.

Text classification. SST-2 is a binary single-sentence classification task consisting of movie reviews with corresponding sentiment [121]. MRPC is a collection of sentence pairs from online news with labels suggesting whether the pair is semantically equivalent

	SST-2	MRPC	POS	NER
DeepJudge-JSD-Soft:				
Negative Suspect	(0.012, 0.032)	(0.009, 0.161)	(0.016, 0.444)	(0.001, 0.416)
Positive Suspect	(0.001, 0.002)	(0.001, 0.002)	(0.087, 0.279)	(0.002, 0.201)
DeepJudge-JSD-Hard:				
Negative Suspect	(0.013, 0.029)	(0.008, 0.154)	(0.010, 0.432)	(0.009, 0.274)
Positive Suspect	(0.004, 0.005)	(0.003, 0.007)	(0.029, 0.112)	(0.011, 0.052)
DRW-P_{snr}-Soft:				
Negative Suspect	(0.008, 4.775)	(0.128, 2.607)	(0.012, 2.309)	(0.105, 4.243)
Positive Suspect	(18.82, 25.77)	(17.81, 24.25)	(20.59, 28.73)	(17.25, 25.22)
DRW-P_{snr}-Hard:				
Negative Suspect	(0.011, 4.235)	(0.012, 3.678)	(0.182, 2.869)	(0.203, 4.183)
Positive Suspect	(16.38, 22.77)	(16.70, 21.80)	(16.23, 25.67)	(16.19, 25.49)

Table 4.2: The probing results for DeepJudge and DRW in soft distillation and hard distillation settings. We present the range of JSD and P_{snr}. The first value in parentheses is the minimum score and the second value is the maximum score. A larger gap in score between the negative and positive suspect models indicates that the detection method performs better in identifying the extracted model.

or not [122]. We use the final hidden vector of the special `[CLS]` token of BERT as the input to a linear layer, which serves as the sentence classifier. The ID of the second token in the sentence is set as the input x for the hash function $g(\cdot)$. Since GLUE does not include any test dataset, we use accuracy of the validation set as the evaluation metric.

For each task, we train the protected model to achieve the best performance on the validation set. As demonstrated in Table 4.1, the victim model has comparable performance to BERT [98]. For soft and hard label distillation, we split the training data in each task into two parts and use the first half to query the victim model. Then the extracted model is trained for 20 epochs on the pseudo-labeled dataset. We choose the same key $K = (c^*, f_w, \mathbf{v}_k, \mathbf{v}_s, \mathbf{M})$, where frequency $f_w = 16.0$, watermark level $\varepsilon = 0.2$ and $\{\mathbf{v}_k, \mathbf{v}_s, \mathbf{M}\}$ are generated with different random seed. We set target class $c^* = 22$ ("NNP" tag) for POS Tagging, $c^* = 2$ ("I-PER" tag) for NER and $c^* = 0$ ("negative"

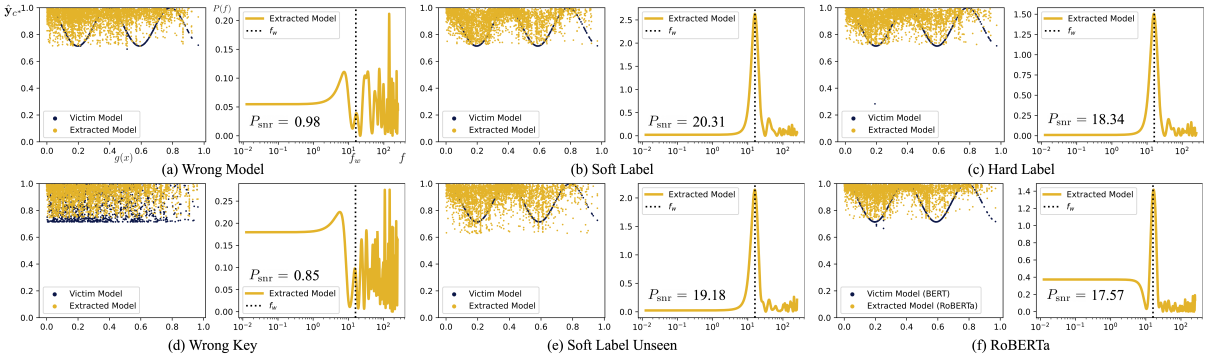

Figure 4.2: Examples of DRW in NER task. The left panel of each sub-figure plots the output of the target class c^* for the victim model and the extracted model ($\hat{\mathbf{y}}_{c^*}$ vs. $g(x)$). The right panel of each sub-figure plots the power spectrum value for output of the extracted model($P(f)$ vs. f). We also display the P_{snr} value for signal strength of the extracted model.

class) for SST-2/MRPC. We set data selection ratio $\tau = 0.5$ to add watermarks to half of the output data. More details for the experiment setting can be found in Appendix 4.5.4.

Baseline We take the state-of-the-art method DeepJudge [123] as a baseline against DRW. DeepJudge quantitatively tests the similarities between the victim model and suspect model, then determines whether the suspect model is a copy based on the testing metrics. Since DeepJudge is designed for continuous signals such as images and audio, we modify the method to apply it to texts. We consider the black-box setting for DeepJudge, and compute Jensen-Shanon Distance (JSD) [124] for the probing dataset of the victim model and the extracted model. JSD measures the similarity of two probability distributions. We use the probing dataset to query both the victim model and the suspect model, and then calculate JSD of the output layer as follows

$$JSD\left(\hat{\mathbf{y}}, \hat{\mathbf{q}}\right) = \frac{1}{2|\mathcal{D}_p|} \sum_{x \in \mathcal{D}_p} K\left(\hat{\mathbf{y}}(x), u\right) + K\left(\hat{\mathbf{q}}(x), u\right)$$

where $u = (\hat{\mathbf{y}}(x) + \hat{\mathbf{p}}(x))/2$ and $K(\cdot, \cdot)$ is the Kullback-Leibler divergence. A small JSD value implies similar output distribution of the two models, which further indicates that the suspect model may be distilled from the victim model.

Evaluation We evaluate the performance of the victim model and the extracted model with accuracy/F1 score. In order to compare DRW with DeepJudge in detecting extracted models, we can reduce this binary classification problem to thresholding a particular test score. Since DRW and DeepJudge use different scores to detect the extracted model, we set up a series of ranking tasks to show the effect of these scores. For each task, we train 10 extracted models from the watermarked victim model with different random initialization as positive samples, 10 extracted models from the unwatermarked victim model with different random initialization, and 10 models from scratch with true labels as negative samples. For DRW, we use the watermark signal strength values P_{snr} as the score for ranking (identifying whether it is an extracted model); for DeepJudge, we use JSD as the score. Next, we compute the mean average precision (mAP) for the ranking tasks which assesses the model extraction detection performance. A higher mAP means the detecting method can distinguish the victim and the suspect model better.

We show the experiment results in the following subsections.

4.4.2 Effectiveness: Is DRW able to identify model infringements?

We evaluate our method in two settings, distillation with soft labels and distillation with hard labels. The results are displayed in Table 4.1. DeepJudge performs well on SST-2 and MRPC tasks but it can not effectively detect the extracted models in POS Tagging and NER tasks. In contrast, our method can successfully detect the extraction

with 100% mAP across all tasks in both settings. We also present the range of JSD and P_{snr} in Table 4.2. Regarding the performance of DeepJudge on POS Tagging and NER tasks, the JSD intervals for positive and negative samples overlap each other, resulting in the aforementioned lower mAP compared to DRW. A case in point is DeepJudge-JSD-Hard for NER task, where the ranges for negative suspect score and positive suspect score are $[0.009, 0.274]$ and $[0.011, 0.052]$ respectively. The overlapping intervals lead to the imperfect detection result, i.e., mAP $= 0.40$. Whereas, DRW is able to *perfectly* distinguish between positive and negative suspects. Typically, P_{snr} for the negative suspect is smaller than 5 while that for the positive suspect is larger than 15.

4.4.3 Fidelity: Does DRW decrease the performance of the model?

The results for the model performance at the watermark level $\varepsilon = 0.2$ are displayed in Table 4.1. The perturbed API (victim model with argmax soft/sampling hard) only has a slight performance drop (within 5%) in comparison to the original one due to the trade-off between detection effectiveness and model performance. For the victim model API, argmax soft exhibits less performance drop than the sampling hard, since the argmax of the soft label remains unchanged with small perturbation. For the extracted model, distillation with soft label tends to have a better accuracy/F1 score than that with hard label. Additionally, the performances of extracted models are very close to those of victim models, a clear manifestation of the distillation success.

4.4.4 Case Study

We present how our method works on some examples in NER task. We fix the victim model and choose different settings for the suspect model across all the examples. For

the watermarked ones, we set $f_w = 16, \varepsilon = 0.2$ and $c^* = 2$.

In Figure 4.2 (a), we show how DRW works on a suspect model that does not extract the victim model. We select a model trained from scratch with true labels as a negative example. There is no sinusoidal signal in the output of the suspect model hence a small P_{snr}.

In Figure 4.2 (b)(c), we illustrate the effect on soft distillation and hard distillation. We use the watermark key K to extract the output of the victim model and suspect model. The extracted model clearly follows the victim model and there is a prominent peak at frequency f_w. Note that suspect model distillation with soft labels has a higher P_{snr} than the one with hard labels. This is because the training process of extracted models can be more effective and faster with soft labels [110].

In Figure 4.2 (d), we validate the *secrecy* of our method. If the adversary does not have the secret key, it can not justify what the watermark is or whether there exist watermarks. The output of the victim model and extracted model are almost indiscernible when we use a wrong key to project them given the hash function $g(\cdot)$.

In Figure 4.2 (e)(f), we demonstrate the generality of our method. Watermarking algorithm should be independent of the dataset and the ML algorithms. In sub-figure (e), a different dataset is used to probe the suspect model. To be specific, we select the second half of the training data as the probing dataset, rather than the first half used in previous experiments. The results imply that DRW turns out to work well when we use unseen data to produce the probing dataset for the suspect model. In sub-figure (f), we choose a different backbone RoBERTa [99] for the suspect model, in which the victim model continues to be the BERT model. The high peak in the power spectrum at frequency f_w reveals that DRW is still able to detect the signal.

Ablation Study

4.4.5 Does watermark level impact detection?

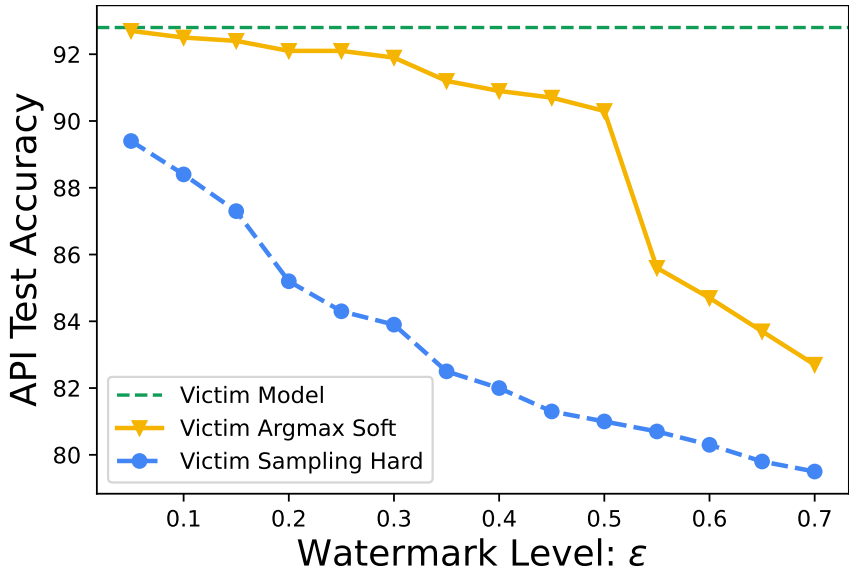

Figure 4.3: Test accuracy of victim model API with different watermark level in SST-2 task.

An important aspect of watermarking is how much perturbation we add to the output of the victim model. Theoretically, a smaller watermark level is associated with a higher accuracy/F1 score of the victim, yet it makes it harder to extract the signal from the probing results. We conduct two experiments to investigate the effect of the watermark level.

In the first experiment, we vary the watermark level in SST-2 task. According to the Theorem 4.3.1, the accuracy of the victim model output is bounded and a higher watermark level causes poorer performance. As shown in Figure 4.3, when the watermark level rises from 0 to 0.7, the performance drops by around 10 percent. It is worth noting that a big drop of the argmax soft emerges as ε passes 0.5, which means the argmax of the output is highly likely to be changed in this case.

In the second experiment, we design 10 sets of ranking tasks, and build up 10 positive

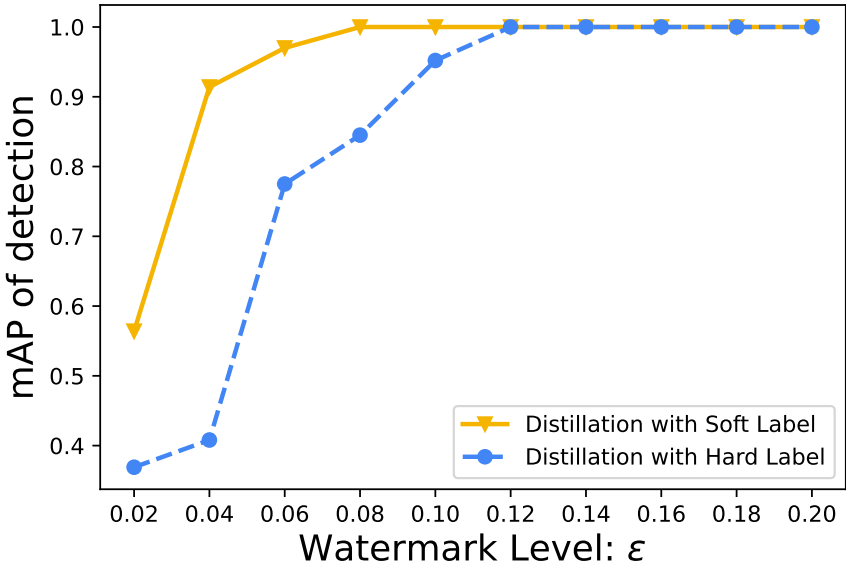

Figure 4.4: Model detection results with different watermark levels in NER task.

samples together with 20 negative samples (similar to the setting in Section 4.4.1) for each set in NER task. The watermark level is the only varied parameter across different tasks, ranging from 0.02 to 0.2. We plot the mAP of the detection against the watermark level in Figure 4.4. When the watermark level ε is below 0.12, DRW can not generate perfect detection of positive and negative suspects, indicating that the adversary may not convey a strong sinusoidal signal at a low watermark level. In this case, DRW can not extract the watermark in frequency space and thus fails to detect it successfully.

These two experiments demonstrate the trade-off between the detection effectiveness and the victim model's performance after watermarking.

4.4.6 Do categories affect watermark protection?

We vary the target class c^* of the watermark key K in POS Tagging task. We add watermarks to four different categories and then train the extracted model by soft distillation and hard distillation. The results of the signal-to-noise ratio P_{snr} are visualized in Figure 4.5. The effect of the watermark will be more salient if the category involves

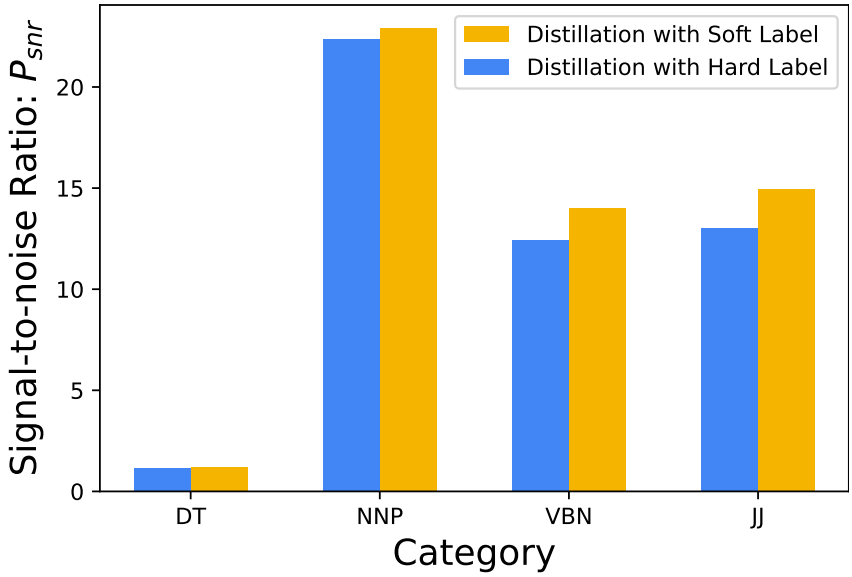

Figure 4.5: Adding watermark to four categories in POS Tagging task. "DT": determiner; "NNP": proper noun, singular; "VBN": verb, past practice; "JJ": adjective.

more samples. Since "NNP" covers the most (14.16%) of all tokens, adding watermark to "NNP" produces the strongest signal. In contrast, the determiners ("DT") category only has a few number of types, such as "the" and "a". As a result, adding watermark to "DT" is ineffective as it is hard to add a periodic signal to a very discrete domain.

4.4.7 How much should be selected for watermarking?

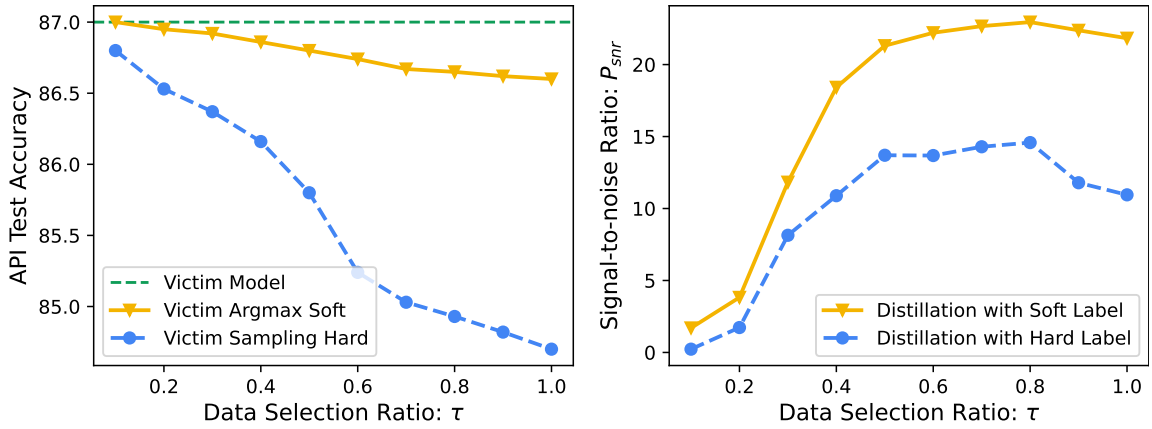

Figure 4.6: Output accuracy of the victim model and signal strength of the extracted model with different data selection ratio τ in MRPC task.

A critical design of our method is that we apply selection vector $\mathbf{v}_s$ to select a portion of the victim model output to be watermarked. We change the ratio of the watermarked data by tuning the data selection ratio τ in MRPC task. The results shown in Figure 4.6 indicate that the accuracy of the victim model output falls with a higher data selection ratio, yet it introduces a greater signal strength of the extracted model. This trade-off is similar to the one described in Section 4.4.5. 0.5 could be a reasonable selection ratio.

4.5 Conclusion

In this work, we propose **D**istillation-**R**esistant **W**atermarking (DRW), a novel and unified watermarking technique against model extraction attacks on NLP models. By injecting watermarks into the prediction output of the victim model, the model owner can detect the watermark if the adversary distills the protected model. We prove the theoretical guarantee of DRW and show remarkable empirical results on text classification and sequence labeling tasks.

Limitations

1) The watermark detection does not work well when the watermarked data covers only a small amount of the whole training data for the extracted model. 2) Our method may not work well when the adversary only makes a few queries to the victim model APIs and trains the extracted model with few-shot learning. 3) If the victim model outputs soft labels, even with watermarking, the adversary can take argmax operation to erase the watermark. So it is better to combine watermarks with hard label output in real-world applications.

Additional Results and Proofs

4.5.1 Proof for the Theorem 1

Theorem 4.5.1 (Restate Theorem 4.3.1)**.** *Without loss of generality, set target class* $c^* = 1$*, so that* $\hat{p} = \hat{\mathbf{p}}_1(x), \hat{y} = \hat{\mathbf{y}}_1, z(x) = \mathbf{z}_1(x)$*. Assume* $\hat{p}(x)$ *is calibrated, i.e.,* $\mathbb{E}[y|\hat{p}(x) = a] = a,\ \forall 0 \leq a \leq 1$*, the argmax soft label of the victim model is* $\hat{y}_s = \mathbf{1}\{\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon} > 0.5\}$ *and the sampling hard label of the victim output is* $\hat{y}_h \sim \text{Ber}(\frac{\hat{p}(x)+\varepsilon(1+z(x))}{1+2\varepsilon})$*. For a fixed* $\mathbf{v}_k$*, given that* $z(x) = \cos\left(f_w g(\mathbf{v}_k, x, \mathbf{M})\right) \in [-1, 1]$ *and the data selection ratio is set to* τ*, then DRW argmax soft label and sampling hard label satisfy:*

$$\begin{aligned}\mathbb{E}_{\mathbf{v}_k}\left[Acc(Argmax\ Soft)\right] &\geq Acc(Victim) \\ &- \tau(0.5+\varepsilon)\mathbb{P}[0.5-\varepsilon \leq \hat{p} \leq 0.5+\varepsilon], \qquad (4.8)\\ \mathbb{E}_{\mathbf{v}_k}\left[Acc(Sampling\ Hard)\right] &\geq (1-\tau)Acc(Victim) \\ &+ \frac{\tau}{1+2\varepsilon}\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]. \qquad (4.9)\end{aligned}$$

Proof. We first prove the argmax soft label case with $\tau = 1$.

$$
\begin{aligned}
&\mathbb{E}\left[\mathbf{1}(\hat{y}_s = y)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = y|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = 1, y = 1|x) + \mathbb{P}(\hat{y}_s = 0, y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y}_s = 1|x)\mathbb{P}(y = 1|x) + \mathbb{P}(\hat{y}_s = 0|x)\mathbb{P}(y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbf{1}\{\hat{p} + \varepsilon z(x) > 0.5\}\mathbb{P}(y = 1|x) + \mathbf{1}\{\hat{p} + \varepsilon z(x) \leq 0.5\}\left(1 - \mathbb{P}(y = 1|x)\right)\right] \\
=&\mathbb{E}\Big[\mathbb{E}\left[\mathbf{1}\{\hat{p} + \varepsilon z(x) > 0.5\}\mathbb{P}(y = 1|x) + \mathbf{1}\{\hat{p} + \varepsilon z(x) \leq 0.5\}\left(1 - \mathbb{P}(y = 1|x)\right)|\hat{p}\right]\Big] \\
\geq&\mathbb{E}\Big[\mathbb{E}\left[\mathbf{1}\{\hat{p} - \varepsilon > 0.5\}\mathbb{P}(y = 1|x)|\hat{p}\right] + \mathbb{E}\left[\mathbf{1}\{\hat{p} + \varepsilon \leq 0.5\}(1 - \mathbb{P}(y = 1|x))|\hat{p}\right]\Big] \\
=&\mathbb{E}\Big[\mathbf{1}\{\hat{p} - \varepsilon > 0.5\}\mathbb{E}\left[\mathbb{P}(y = 1|x)|\hat{p}\right] + \mathbf{1}\{\hat{p} + \varepsilon \leq 0.5\}\mathbb{E}\left[1 - \mathbb{P}(y = 1|x)|\hat{p}\right]\Big] \\
=&\mathbb{E}\Big[\mathbf{1}\{\hat{p} > 0.5 + \varepsilon\}\hat{p} + \mathbf{1}\{\hat{p} \leq 0.5 - \varepsilon\}(1 - \hat{p})\Big] \\
=&\underbrace{\mathbb{E}\left[\mathbf{1}\{\hat{p} > 0.5\}\hat{p} + \mathbf{1}\{\hat{p} \leq 0.5\}(1 - \hat{p})\right]}_{\text{Accuracy of victim model without watermark}} - \mathbb{E}\left[\mathbf{1}\{0.5 < \hat{p} \leq 0.5 + \varepsilon\}\hat{p}\right] \\
&+ \mathbb{E}\left[\mathbf{1}\{0.5 - \varepsilon \leq \hat{p} \leq 0.5\}(1 - \hat{p})\right] \\
\geq&\text{Acc(Victim Model)} - (0.5 + \varepsilon)\mathbb{P}(0.5 - \varepsilon \leq \hat{p} \leq 0.5 + \varepsilon)
\end{aligned}
$$

where the first "$\geq$" follows from $|z(x)| \leq 1$; the third "=" follows from the conditional independence of $\hat{y}_s$ and y given x; the seventh "=" follows from the calibration assumption, i.e. $\mathbb{E}\left[\mathbb{P}(y = 1|x)|\hat{p}(x)\right] = \hat{p}(x)$.

Notice that over the distribution of $\mathbf{v}_s$ selects every unique x with probability τ independently to everything else, by exchanging the order of expectation, it is easy to prove that the expected accuracy is a convex combination of the accuracy of the victim model (with weight $1 - \tau$) and the case above (with weight τ). This completes the proof for argmax soft label.

We then start by analyzing the sampling hard label case with $\tau = 1$.

$$
\begin{aligned}
&\mathbb{E}\left[\mathbf{1}(\hat{y} = y)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y} = y|x)\right] \\
=&\mathbb{E}\left[\mathbb{P}(\hat{y} = 1, y = 1|x) + \mathbb{P}(\hat{y} = 0, y = 0|x)\right] \\
=&\mathbb{E}\left[\mathbb{E}(\hat{y}|x)\mathbb{E}(y|x) + \mathbb{E}(1-\hat{y}|x)\mathbb{E}(1-y|x)\right] \\
=&\mathbb{E}\left[\left(\frac{\hat{p}}{1+2\varepsilon} + \frac{\varepsilon(1+z(x))}{1+2\varepsilon}\right)\mathbb{E}(y|x)\right. \\
&\left. + \left(\frac{1-\hat{p}}{1+2\varepsilon} + \frac{\varepsilon(1-z(x))}{1+2\varepsilon}\right)\mathbb{E}(1-y|x)\right] \\
=&\frac{1}{1+2\varepsilon}\underbrace{\mathbb{E}\left[\hat{p}\mathbb{E}(y|x) + (1-\hat{p})\mathbb{E}(1-y|x)\right]}_{A} \\
&+ \frac{\varepsilon}{1+2\varepsilon}\underbrace{\mathbb{E}\left[(1+z(x))\mathbb{E}(y|x) + (1-z(x))\mathbb{E}(1-y|x)\right]}_{B}
\end{aligned}
$$

$$
\begin{aligned}
A =&\mathbb{E}\left[\mathbb{E}\left[\hat{p}\mathbb{E}(y|x) + (1-\hat{p})\mathbb{E}(1-y|x)|\hat{p}\right]\right] \\
=&\mathbb{E}\left[\hat{p}\mathbb{E}(y|\hat{p}) + (1-\hat{p})\mathbb{E}(1-y|\hat{p})\right] \\
=&\mathbb{E}\left[\hat{p}^2 + (1-\hat{p})^2\right] \\
=&\mathbb{E}\left[2\hat{p}^2 - 2\hat{p} + 1\right]
\end{aligned}
$$

where the third line follows from the calibration assumption, i.e., $\mathbb{E}[y|\hat{p}(x) = a] = a$.

$$
\begin{aligned}
B =&\mathbb{E}\left[\mathbb{E}(y|x) + \mathbb{E}(y|x)z(x) + 1 - z(x)\right. \\
&\left. - \mathbb{E}(y|x) + \mathbb{E}(y|x)z(x)\right] \\
=&1 + \mathbb{E}\left[\left(2\mathbb{E}(y|x) - 1\right)z(x)\right] \\
\geq&0
\end{aligned}
$$

where the last line follows from the facts that $|z(x)| \leq 1$ and $|2\mathbb{E}(y|x) - 1| \leq 1$.

Finally, notice that for each x the probability to be chosen to add watermark and to sample the output is τ independently, thus the expected accuracy is the convex combination of the accuracy of the victim model and that of the fully watermarked model. □

4.5.2 Distribution Property

Lemma 4.5.2. *Assume* $\mathbf{v} \sim \mathcal{U}(0,1),\ \mathbf{v} \in \mathbb{R}^n$ *and* $\mathbf{x} \sim \mathcal{N}(0,1),\ \mathbf{x} \in \mathbb{R}^n$*, where* $\mathbf{v}$ *and* $\mathbf{x}$ *are both i.i.d. and independent of each other. Then we have:*

$$\frac{1}{\sqrt{n}}\mathbf{v} \cdot \mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right),\ n \to \infty$$

Proof. Let $u_i = \mathbf{v}_i\mathbf{x}_i,\ i \in 1, 2, \ldots, n$. By assumption, u_i are *i.i.d.*. Clearly, the first and second moments are bounded, so the claim follows from the classical central limit theorem,

$$\sqrt{n}\bar{u}_n = \frac{\sum_{i=1}^{n} u_i}{\sqrt{n}} \rightsquigarrow \mathcal{N}\left(\mu, \sigma^2\right) \quad \text{as} \quad n \to \infty$$

where

$$\begin{aligned}
\mu &= \mathbb{E}\left(u_i\right) = \mathbb{E}\left(\mathbf{v}_i\mathbf{x}_i\right) = \mathbb{E}\left(\mathbf{v}_i\right)\mathbb{E}\left(\mathbf{x}_i\right) \\
&= 0 \\
\sigma^2 &= \mathrm{Var}(u_i) = \mathbb{E}\left(u_i^2\right) - \left(\mathbb{E}\left(u_i\right)\right)^2 \\
&= \mathbb{E}\left(u_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2\mathbf{x}_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2\right)\mathbb{E}\left(\mathbf{x}_i^2\right) \\
&= \frac{1}{3}
\end{aligned}$$

It follows that given large n

$$\frac{1}{\sqrt{n}}\mathbf{v}\cdot\mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right)$$

□

4.5.3 Modified Softmax Properties

Lemma 4.5.3 (Lemma 1 in [107]). *Let $\hat{\mathbf{p}}$ be the softmax output of a model $\mathcal{V}$, then the modified softmax $\hat{\mathbf{y}}$, as defined in Equation 4.4 satisfies $0 \leq \hat{\mathbf{y}}_i \leq 1$ and $\sum_{i=1}^{m} \hat{\mathbf{y}}_i = 1$.*

Proof. Notice that in Equation 4.4, when $g(\mathbf{v}_s, x, \mathbf{M}) > \tau$, $\hat{\mathbf{y}} = \hat{\mathbf{p}}$, so that it satisfies the property above.

By the definition of softmax, for all class $c \in \{1, \ldots, m\}$ we have

$$0 \leq \hat{\mathbf{p}}_c \leq 1, -1 \leq \mathbf{z}_c(x) \leq 1.$$

Therefore, when $c = c^*$, we have

$$0 \leq \hat{\mathbf{p}}_c + \varepsilon\left(1 + \mathbf{z}_c(x)\right) \leq 1 + 2\varepsilon,$$

and then

$$0 \leq \frac{\hat{\mathbf{p}}_c + \varepsilon\left(1 + \mathbf{z}_c(x)\right)}{1 + 2\varepsilon} \leq 1.$$

When $c \neq c^*$, since $m \geq 2$, we have

$$0 \leq \hat{\mathbf{p}}_c + \frac{\varepsilon\left(1 + \mathbf{z}_c(x)\right)}{m - 1} \leq 1 + \frac{2\varepsilon}{m - 1} \leq 1 + 2\varepsilon$$

and then

$$0 \leq \frac{\hat{\mathbf{p}}_c + \frac{\varepsilon(1+\mathbf{z}_c(x))}{m-1}}{1+2\varepsilon} \leq 1.$$

Thus, $\hat{\mathbf{q}}$ satisfies $0 \leq \hat{\mathbf{y}}_i \leq 1$.

To prove $\sum_{i=1}^{m} \hat{\mathbf{y}}_i = 1$, we use the fact that $\mathbf{z}_{c^*} + \mathbf{z}_{i \neq c^*} = 0$ and obtain

$$\begin{aligned}
\sum_{i=1}^{c} \hat{\mathbf{y}}_i &= \frac{\hat{\mathbf{p}}_{c^*} + \varepsilon\left(1 + \mathbf{z}_{c^*}\right)}{1+2\varepsilon} + \sum_{i \neq c^*} \frac{\hat{\mathbf{p}}_i + \frac{\varepsilon(1+\mathbf{z}_i)}{m-1}}{1+2\varepsilon} \\
&= \sum_{i=1}^{m} \frac{\hat{\mathbf{p}}_i}{1+2\varepsilon} + \sum_{i \neq c^*} \frac{\varepsilon\left(1 + \mathbf{z}_{c^*} + 1 + \mathbf{z}_i\right)}{(m-1)(1+2\varepsilon)} \\
&= \frac{1}{1+2\varepsilon} + \frac{2\varepsilon}{1+2\varepsilon} \\
&= 1
\end{aligned}$$

□

4.5.4 Experiment Details

We provide more details for the experiments in this section.

We build our classification models upon `bert-base-uncased` from Hugging Face[2]. The model contains 110M parameters. We add a dropout layer before the last linear layer with a dropout rate of 0.5. We implement DRW in PyTorch 1.11.0 on a server with 4 NVIDIA TITAN-Xp GPUs. We set batch size to 8 for SST-2 and MRPC tasks, and 32 for POS Tagging and NER tasks.

We train the victim model using AdamW [125] optimizer with learning rate 1e-5 and epsilon 1e-8. Each victim model is trained 40 epochs and the one with the best validation results is chosen.

[2]https://huggingface.co/

Regarding the extracted model, we use half of the training data to query the victim model and obtain the labeled dataset. Then the extracted model is trained with Adam [126] optimizer for 20 epochs with learning rate 5e-5. The average training time is 3 minutes for each epoch.

We show the results for RoBERTa model in Section 4.4.4. In this setting, we choose `roberta-base` from Hugging Face, which has 125M parameters.

Chapter 5

Language Generation Model Watermark

Language generation models have been an increasingly powerful enabler for many applications. Many such models offer free or affordable API access, which makes them potentially vulnerable to model extraction attacks through distillation. To protect intellectual property (IP) and ensure fair use of these models, various techniques such as lexical watermarking and synonym replacement have been proposed. However, these methods can be nullified by obvious countermeasures such as "synonym randomization". To address this issue, we propose GINSEW, a novel method to protect text generation models from being stolen through distillation. The key idea of our method is to inject secret signals into the probability vector of the decoding steps for each target token. We can then detect the secret message by probing a suspect model to tell if it is distilled from the protected one. Experimental results show that GINSEW can effectively identify instances of IP infringement with minimal impact on the generation quality of protected APIs. Our method demonstrates an absolute improvement of 19 to 29 points on mean average precision (mAP) in detecting suspects compared to previous methods against watermark

removal attacks. Our code is available at https://github.com/XuandongZhao/Ginsew.

5.1 Introduction

Large language models (LLMs) have become increasingly powerful [127, 51], but their owners are reluctant to open-source them due to high training costs. Most companies provide only API access to their models for free or for a fee to cover innovation and maintenance costs. While many applications benefit from these APIs, some are looking for cheaper alternatives.

While healthy competition is good for preventing monopoly in the LLM service market, a fairly priced API-service may expose a "short cut" that allows a company to distill a comparable model at a much lower cost — forcing the original model creator out of the market and stifling future innovation. Anticipating this, rational LLM owners will likely never provide API access at a fair price. Instead, they must dramatically increase fees - far beyond service costs - to make model distillation unprofitable.

The increased fee might be tolerable for downstream applications with good commercial values, but it makes research and LLM applications for public good unaffordable. To address this, governments could fund public LLMs for research and public service. However, public LLMs could also be distilled for commercial gain or military use by unethical entities.

Recently, a Stanford group [73] claimed to have distilled ChatGPT for just $600 in API calls, demonstrating the issue's urgency. The cheaper cost to distill versus retrain models can be justified theoretically [128]: The sample complexity for training a model from raw data to ϵ-excess risk is on the order of $O(1/\epsilon^2)$; whereas model distillation that uses model-generated labels operates in a "realizable" regime, thus requires only $O(1/\epsilon)$ samples.

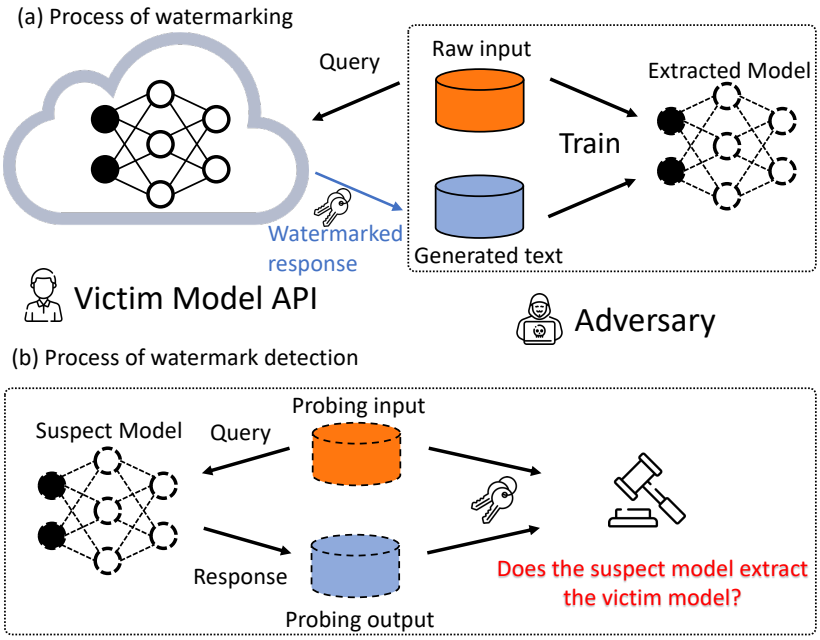

Figure 5.1: Overview of the process of watermarking and the process of watermark detection. The victim model API embeds watermarks in the response to input queries from the adversaries. The API owner can then use a key to verify if the suspect model has been distilled from the victim model.

This brings us to the central question: How can we prevent model-stealing attacks through distillation?

It appears to be a mission impossible. By providing the state-of-the-art LLM service, the information needed to replicate it has been encoded in the provided API outputs themselves. Researchers decided to give up on preventing learnability, but instead, try to watermark the API outputs so models trained using the watermarked outputs can be traced to the original model. Recent works [129, 130] use a trigger set to embed invisible watermarks on the neurons of the commercial models before distribution. When model theft is suspected, model owners can conduct an official ownership claim with the aid of the trigger set. Existing works [117, 131] use a synonym replacement strategy to add surface-level watermarks to a text generation model. However, these methods can be bypassed by an adversary randomly replacing synonyms in the output and removing

watermarks from the extracted model, making the protection ineffective.

In this chapter, we propose generative invisible sequence watermarking (GINSEW), a method to protect text generation models and detect stolen ones. Figure 5.1 illustrates the overall process. The core idea is to inject a secret sinusoidal signal into the model's generation probabilities for words. This signal does not harm the model's generation quality. To detect whether a candidate model is stealing from the target model, GINSEW identifies the sinusoidal frequency and compares it with the secret key. GINSEW provides a more robust mechanism for protecting the intellectual property of the model, even under random synonym replacement attacks.

The contributions of this chapter are as follows:

- We propose generative invisible sequence watermarking (GINSEW), a method to protect text generation models against model extraction attacks with invisible watermarks.
- We carry out experiments on machine translation and story generation on a variety of models. Experimental results show that our method GINSEW outperforms the previous methods in both generation quality and robustness of IP infringement detection ability. Even with adversarial watermark removal attacks, GINSEW still gains a significant improvement of 19 to 29 points in mean average precision (mAP) of detecting suspects.

5.2 Related Work

Model extraction attacks. Model extraction attacks, also known as model inversion or model stealing, pose a significant threat to the confidentiality of machine learning models [132, 133, 101, 134]. These attacks aim to imitate the functionality of a black-box

victim model by creating or collecting a substitute dataset. The attacker then uses the victim model's APIs to predict labels for the substitute dataset. With this pseudo-labeled dataset, the attacker can train a high-performance model that mimics the victim model and can even mount it as a cloud service at a lower price. This type of attack is closely related to knowledge distillation (KD) [109], where the attacker acts as the student model that approximates the behavior of the victim model. In this chapter, we specifically focus on model extraction attacks in text generation. Previous works [101, 135] have shown that adversaries can use sequence-level knowledge distillation to mimic the functionality of commercial text generation APIs, which poses a severe threat to cloud platforms.

Watermarking. Watermarking is a technique used to embed unseen labels into signals such as audio, video, or images to identify the owner of the signal's copyright. In the context of machine learning models, some studies [136, 129, 130] have used watermarks to prevent exact duplication of these models, by inserting them into the parameters of the protected model or constructing backdoor images that activate specific predictions. However, protecting models from model extraction attacks is difficult as the parameters of the suspect model may be different from those of the victim model and the backdoor behavior may not be transferred.

Watermarking against model extraction. Recently, several studies have attempted to address the challenge of identifying extracted models that have been distilled from a victim model, with promising results in image classification [137, 138] and text classification [8]. CosWM [137] embeds a watermark in the form of a cosine signal into the output of the protected model. This watermark is difficult to eliminate, making it an effective way to identify models that have been distilled from the protected model. Nonetheless, CosWM only applies to image classification tasks. As for text generation, [117]

Algorithm 8 Watermarking process

Inputs: Input text $\boldsymbol{x}$, probability vector $\mathbf{p}$ from the decoder of the victim model, vocab $\mathcal{V}$, group 1 $\mathcal{G}_1$, group 2 $\mathcal{G}_2$, hash function $g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})$.
Output: Modified probability vector $\mathbf{p}$
Calculate probability summation of tokens in group 1 and group 2: $Q_{\mathcal{G}_1} = \sum_{i \in \mathcal{G}_1} \mathbf{p}_i,\ Q_{\mathcal{G}_2} = \sum_{i \in \mathcal{G}_2} \mathbf{p}_i$
Calculate the periodic signal

$$z_1(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})\right),$$
$$z_2(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M}) + \pi\right)$$

Set $\tilde{Q}_{\mathcal{G}_1} = \frac{Q_{\mathcal{G}_1} + \varepsilon(1+z_1(\boldsymbol{x}))}{1+2\varepsilon}, \tilde{Q}_{\mathcal{G}_2} = \frac{Q_{\mathcal{G}_2} + \varepsilon(1+z_2(\boldsymbol{x}))}{1+2\varepsilon}$
for $i = 1$ **to** $|\mathcal{V}|$ **do**
    if $i \in \mathcal{G}_1$ **then** $\mathbf{p}_i \leftarrow \frac{\tilde{Q}_{\mathcal{G}_1}}{Q_{\mathcal{G}_1}} \cdot \mathbf{p}_i$
    else $\mathbf{p}_i \leftarrow \frac{\tilde{Q}_{\mathcal{G}_2}}{Q_{\mathcal{G}_2}} \cdot \mathbf{p}_i$
end for
return p

propose a lexical watermarking method to identify IP infringement caused by extraction attacks. This method involves selecting a set of words from the training data of the victim model, finding semantically equivalent substitutions for them, and replacing them with the substitutions. Another approach, CATER [131], proposes conditional watermarking by replacing synonyms of some words based on linguistic features. However, both methods are surface-level watermarks. The adversary can easily bypass these methods by randomly replacing synonyms in the output, making it difficult to verify by probing the suspect models. GINSEW directly modifies the probability distribution of the output tokens, which makes the watermark invisible and provides a more robust mechanism for identifying extracted models.

5.3 Proposed Method: Ginsew

5.3.1 Problem setup

Our goal is to protect text generation models against model extraction attacks. It enables the victim model owner or a third-party arbitrator to attribute the ownership of a suspect model that is distilled from the victim API. We leverage the secret knowledge that the extracted model learned from the victim model as a signature for attributing ownership.

In a model extraction attack, the adversary, denoted as $\mathcal{S}$, only has black-box access to the victim model's API, denoted as $\mathcal{V}$. The adversary can query $\mathcal{V}$ using an auxiliary unlabeled dataset, but can only observe the text output and not the underlying probabilities produced by the API. As a result, the adversary receives the output of $\mathcal{V}$ as generation output or pseudo labels. The attacker's goal is to employ sequence-level knowledge distillation to replicate the functionality of $\mathcal{V}$. The model extraction attack is depicted in Figure 5.1(a).

Our method does not aim to prevent model extraction attacks as we cannot prohibit the adversary from mimicking the behavior of a common user. Instead, we focus on verifying whether a suspect model has been trained from the output of the victim model API. If a suspect model distills well, it carries the signature inherited from the victim. Otherwise, a suspect may not carry such a signature, and its generation quality will be noticeably inferior to the victim.

During verification, we assume the presence of a third-party arbitrator (e.g., law enforcement) with white-box access to both the victim and suspect models, as well as a probing dataset. The arbitrator or model owner compares the output of the suspect model to the secret signal in the watermark with a key: if the output matches the watermark, the suspect model is verified as a stolen model.

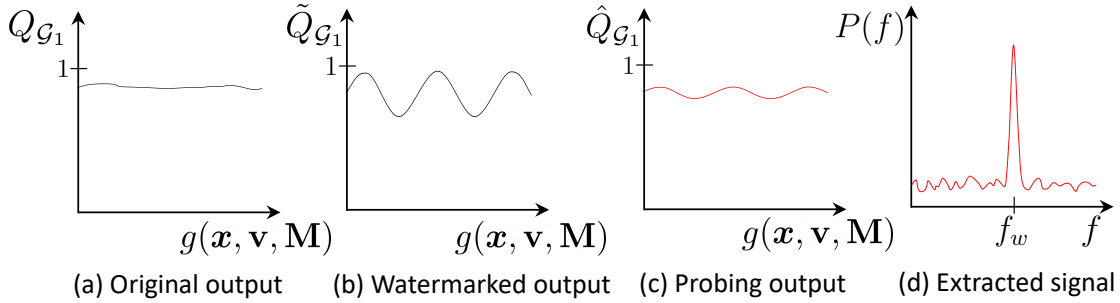

Figure 5.2: The process of GINSEW. (a) The original group probability of the victim model is represented by $Q_{\mathcal{G}_1}$. (b) The API owner applies a sinusoidal perturbation to the predicted group probability, resulting in a watermarked output, denoted as $\tilde{Q}_{\mathcal{G}_1}$. (c) If the adversary attempts to distill the victim model, the extracted model will convey this periodical signal. (d) After applying a Fourier transform to the output with a specific key, a peak in the frequency domain at frequency f_w can be observed.

5.3.2 Generative invisible sequence watermarking

GINSEW dynamically injects a watermark in response to queries made by an API's end-user. It is invisible because it is not added to the surface text. We provide an overview of GINSEW in Figure 5.1, consisting of two stages: i) watermarking stage and ii) watermark detection stage.

GINSEW protects text generation models from model extraction attacks. We achieve this by carefully manipulating the probability distribution of each token generated by the model, specifically by modifying the probability vector during the decoding process. This approach allows for a unique signature to be embedded within the generated text, making it easily identifiable by the owner or a third-party arbitrator, while still maintaining the coherence and fluency of the text. The process of GINSEW is illustrated in Figure 5.2.

We present the watermarking process in Algorithm 8. For each token v in the whole vocabulary $\mathcal{V}$, we randomly assign it to two distinct groups, group 1 $\mathcal{G}_1$ and group 2 $\mathcal{G}_2$, so that each group contains $\frac{|\mathcal{V}|}{2}$ words. $|\mathcal{V}|$ represents the vocabulary size. For each input text $\boldsymbol{x}$, we use a hash function, referred to as $g(\cdot)$, to project it into a scalar. This hash function takes three inputs: input text $\boldsymbol{x}$, phase vector $\mathbf{v} \in \mathbb{R}^n$ and token matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}| \times n}$.

Note that each token corresponds to a row in the matrix. The elements of the phase vector $\mathbf{v}$ are randomly sampled from a uniform distribution $[0, 1)$, while the elements of the token matrix $\mathbf{M}$ are randomly sampled from a standard normal distribution $\mathbf{M}_{ij} \sim \mathcal{N}(0, 1)$. Let $\mathbf{M}_i \in \mathbb{R}^n$ denote the i-th row of matrix $\mathbf{M}$, $\mathbf{v}^\top \mathbf{M}_i \sim \mathcal{N}(0, \frac{n}{3})$ (See proof in Appendix 5.5.2). We then use the probability integral transformation F to obtain a uniform distribution of the hash values: $g(\boldsymbol{x}, \mathbf{v}, \mathbf{M}) = F(\mathbf{v}^\top \mathbf{M}_{\text{tok}(\boldsymbol{x})}) \sim \mathcal{U}(0, 1)$, where $\text{tok}(\boldsymbol{x})$ denotes the ID of the second token in the input text.

We design a periodic signal function based on the input. The hash function $g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})$ returns a scalar determining the amount of noise to be added to the output. As shown in Algorithm 8, given a probability vector $\mathbf{p} \in [0, 1]^{|\mathcal{V}|}$, we calculate the total probability of group 1, $Q_{\mathcal{G}_1} = \sum_{i \in \mathcal{G}_1} \mathbf{p}_i$, and the total probability of group 2, $Q_{\mathcal{G}_2} = \sum_{i \in \mathcal{G}_2} \mathbf{p}_i$. Following [137], we calculate the periodic signal function, where $f_w \in \mathbb{R}$ is the angular frequency.

$$z_1(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})\right), \tag{5.1}$$

$$z_2(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M}) + \pi\right) \tag{5.2}$$

We can obtain a watermark key by combining a set of variables $K = (f_w, \mathbf{v}, \mathbf{M})$. Note that $z_1(\boldsymbol{x}) + z_2(\boldsymbol{x}) = 0$. Next, we compute the periodic signal for the modified group probability

$$\tilde{Q}_{\mathcal{G}_1} = \frac{Q_{\mathcal{G}_1} + \varepsilon\left(1 + z_1(\boldsymbol{x})\right)}{1 + 2\varepsilon}, \tag{5.3}$$

$$\tilde{Q}_{\mathcal{G}_2} = \frac{Q_{\mathcal{G}_2} + \varepsilon\left(1 + z_2(\boldsymbol{x})\right)}{1 + 2\varepsilon} \tag{5.4}$$

We prove that $0 \leq Q_{\mathcal{G}_1}, Q_{\mathcal{G}_2} \leq 1$ and $Q_{\mathcal{G}_1} + Q_{\mathcal{G}_2} = 1$ (In Appendix 5.5.3). ε is the watermark level, measuring how much noise is added to the group probability. Then we change the elements in the probability vector such that they align with the perturbed

Algorithm 9 Watermark detection

Inputs: Suspect model $\mathcal{S}$, sample probing data $\mathcal{D}$ from the training data of $\mathcal{S}$, vocab $\mathcal{V}$, group 1 $\mathcal{G}_1$, group 2 $\mathcal{G}_2$, hash function $g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})$, filtering threshold value $q_{\min}$.
Output: Signal strength
Initialize $\mathcal{H} = \emptyset$
for each input $\boldsymbol{x}$ in $\mathcal{D}$ **do**
  $t = g(\mathbf{v}, \boldsymbol{x}, \mathbf{M})$
  for each decoding step of $\mathcal{S}(\boldsymbol{x})$ **do**
    Get probability vector $\hat{\mathbf{p}}$ from the decoder of the suspect model.
    $\hat{Q}_{\mathcal{G}_1} = \sum_{i \in \mathcal{G}_1} \hat{\mathbf{p}}_i$
    $\mathcal{H} \leftarrow \mathcal{H} \cup (t, \hat{Q}_{\mathcal{G}_1})$
  end for
end for
Filter out elements in $\mathcal{H}$ where $\hat{Q}_{\mathcal{G}_1} \leq q_{\min}$, remaining pairs form the set $\widetilde{\mathcal{H}}$.
Compute the Lomb-Scargle periodogram from the pairs $(t^{(k)}, \hat{Q}^{(k)}_{\mathcal{G}_1}) \in \widetilde{\mathcal{H}}$
Compute P_{snr} in Equation 5.5.
return P_{snr}

group probability. In this way, we are able to embed a hidden sinusoidal signal into the probability vector of the victim model. We can use decoding methods like beam search, top-k sampling, or other methods to generate the output with the new probability vector.

5.3.3 Detecting watermark from suspect models

In order to detect instances of IP infringement, we first create a probing dataset $\mathcal{D}$ to extract watermarked signals in the probability vector of the decoding steps. $\mathcal{D}$ can be obtained from the training data of the extracted model, as the owner has the ability to store any query sent by a specific end-user. Additionally, $\mathcal{D}$ can also be drawn from other distributions as needed.

The process of detecting the watermark from suspect models is outlined in Algorithm 9. For each probing text input, the method first acquires the hash value $t = g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})$. Next, for each decoding step of the suspect model, we add the pair $(t, \hat{Q}_{\mathcal{G}_1})$ to the set $\mathcal{H}$. To eliminate outputs with low confidence, we filter out the pairs with $\hat{Q}_{\mathcal{G}_1} \leq q_{\min}$, where

Algorithm 10 Watermark detection with text alone

Inputs: Suspect model $\mathcal{S}$, sample probing data $\mathcal{D}$ from the training data of $\mathcal{S}$, vocab $\mathcal{V}$, group 1 $\mathcal{G}_1$, group 2 $\mathcal{G}_2$, hash function $g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})$.
Output: Signal strength
Initialize $\mathcal{H} = \emptyset$
for each input $\boldsymbol{x}$ in $\mathcal{D}$ **do**
  $t = g(\mathbf{v}, \boldsymbol{x}, \mathbf{M})$
  $\boldsymbol{y} \leftarrow \mathcal{S}(\boldsymbol{x})$
  for each token of $\boldsymbol{y}$ **do**
    $\mathcal{H} \leftarrow \mathcal{H} \cup (t, \mathbf{1}(\boldsymbol{y}_i \in \mathcal{G}_1))$
  end for
end for
Compute the Lomb-Scargle periodogram from $\mathcal{H}$, and compute P_{snr} in Equation 5.5.
return P_{snr}

the threshold value $q_{\min}$ is a constant parameter of the extraction process. The Lomb-Scargle periodogram [118] method is then used to estimate the Fourier power spectrum $P(f)$, at a specific frequency, f_w, in the probing set $\mathcal{H}$. By applying approximate Fourier transformation, we amplify the subtle perturbation in the probability vector. So that we can detect a peak in the power spectrum at the frequency f_w. This allows for the evaluation of the strength of the signal, by calculating the signal-to-noise ratio P_{snr}

$$\begin{aligned} P_{\text{signal}} &= \frac{1}{\delta} \int_{f_w - \frac{\delta}{2}}^{f_w + \frac{\delta}{2}} P(f) df \\ P_{\text{noise}} &= \frac{1}{F - \delta} \left[\int_0^{f_w - \frac{\delta}{2}} P(f) df + \int_{f_w + \frac{\delta}{2}}^{F} P(f) df \right] \\ P_{\text{snr}} &= P_{\text{signal}} \, / P_{\text{noise}} \, , \end{aligned} \tag{5.5}$$

where δ controls the window width of $\left[f_w - \frac{\delta}{2}, f_w + \frac{\delta}{2}\right]$; F is the maximum frequency, and f_w is the angular frequency embedded into the victim model. A higher P_{snr} indicates a higher peak in the frequency domain, and therefore a higher likelihood of the presence of the secret signal in the suspect model, confirming it distills the victim model.

	IWSLT14			WMT14			ROCStories		
	BLEU ↑	BERTScore ↑	Detect mAP ↑	BLEU ↑	BERTScore ↑	Detect mAP ↑	ROUGE-L ↑	BERTScore ↑	Detect mAP ↑
Original models	34.6	94.2	-	30.8	65.7	-	16.5	90.1	-
Plain watermark									
[117]	33.9	92.7	100	30.5	65.3	100	15.8	89.3	100
CATER [131]	33.8	92.5	76.4	30.5	65.4	78.3	15.6	89.1	83.2
GINSEW	34.2	93.8	100	30.6	65.5	100	16.1	89.6	100
Watermark removed by synonym randomization									
[117]	32.7	90.7	63.1	29.6	64.7	62.3	14.8	88.4	59.6
CATER [131]	32.7	90.6	68.5	29.5	64.7	63.1	14.9	88.4	64.2
GINSEW	33.1	90.9	87.7	29.8	64.9	86.9	15.1	89.0	93.2

Table 5.1: Main results for model performance and detection. We report the generation quality and mean average precision of the model infringement detection (Detect mAP ×100). We use F1 scores of ROUGE-L and BERTScore.

It is worth noting that the threshold and frequency parameters used in the Lomb-Scargle periodogram method can be adjusted to optimize the performance of the proposed method. The specific settings used in our experiments will be discussed in the Experiments section of the chapter. Besides, we present Algorithm 10 demonstrating that we can detect watermarks by analyzing just the generated text itself, without relying on the model's predicted probabilities. A more detailed discussion of this text-only approach can be found in Section 5.4.5.

5.4 Experiments

We evaluate the performance of GINSEW on two common text generation tasks: machine translation and story generation. There are multiple public APIs available for these models[1][2].

Machine translation. In the machine translation task, we utilize the IWSLT14 and WMT14 datasets [139, 140], specifically focusing on German (De) to English (En) translations. We evaluate the quality of the translations based on BLEU [141] and BERTScore

[1]https://translate.google.com/
[2]https://beta.openai.com/overview

[142] metrics. We adopt the official split of train/valid/test sets. BLEU focuses on lexical similarity by comparing n-grams, while BERTScore focuses on semantic equivalence through contextual embeddings. For IWSLT14, a vocabulary consisting of 7,000 BPE [143] units is used, whereas WMT14 employs 32,000 BPE units.

Story generation. For the story generation task, we use the ROCstories [144] corpus. Each story in this dataset comprises 5 sentences, with the first 4 sentences serving as the context for the story and the input to the model, and the 5th sentence being the ending of the story to be predicted. There are 90,000 samples in the train set, and 4081 samples in the validation and test sets. The generation quality is evaluated based on ROUGE [145] and BERTScore metrics. A vocabulary of 25,000 BPE units is used in this task.

Baselines. We compare GINSEW with [117] and CATER [131]. Specifically, [117] propose two watermarking approaches: the first one replaces all the watermarked words with their synonyms; the second one watermarks the victim API outputs by mixing American and British spelling systems. Because the second one is easily eliminated by the adversary through consistently using one spelling system, we focus on their first approach. This method selects a set of words $\mathcal{C}$ from the training data of the victim model. Then for each word $c \in \mathcal{C}$, it finds synonyms for c and forms a set $\mathcal{R}$. Finally, the original words of $\mathcal{C}$ and their substitutions $\mathcal{R}$ are replaced with watermarking words $\mathcal{W}$. As an improvement of [117], CATER proposes a conditional watermarking framework, which replaces the words with synonyms based on linguistic features. The hit ratio is used as the score for the baselines to detect IP infringement

$$\text{hit} = \frac{\#\left(\mathcal{W}_y\right)}{\#\left(\mathcal{C}_y \cup \mathcal{R}_y\right)} \tag{5.6}$$

where $\#(\mathcal{W}_y)$ represents the number of watermarked words $\mathcal{W}$ appearing in the suspect model's output $\boldsymbol{y}$, and $\#(\mathcal{C}_y \cup \mathcal{R}_y)$ is the total number of $\mathcal{C}_y$ and $\mathcal{R}_y$ found in word sequence $\boldsymbol{y}$. We reproduce the watermarking methods with the synonym size of 2 in [117] and CATER. For CATER, we use the first-order Part-of-Speech (POS) as the linguistic feature.

Evaluation. In order to evaluate the performance of GINSEW against the baselines in detecting extracted models, we reduce the binary classification problem into a thresholding task by using a specific test score. Since GINSEW and baselines use different scores to detect the extracted model, we set up a series of ranking tasks to show the effect of these scores. For each task, we train a Transformer [41] base model as the victim model. Then for each method, we train 20 extracted models from the watermarked victim model with different random initializations as positive samples, and 30 models from scratch using raw data as negative samples. For GINSEW, we use the watermark signal strength values P_{snr} (Equation 5.5) as the score for ranking, while for [117] and CATER, we use the hit ratio (Equation 5.6) as the score. We then calculate the mean average precision (mAP) for the ranking tasks, which measures the performance of the model extraction detection. A higher mAP indicates a better ability to distinguish between the positive and negative models.

Experiment setup. For each task, we train the protected models to achieve the best performance on the validation set. As shown in Table 5.1, we train three victim models without watermark on IWSLT14, WMT14 and ROCStories datasets. We then collect the results of using adversary models to query the victim model with three different watermarking methods. By default, we use beam search as the decoding method (beam size = 5). We choose the Transformer base model as the victim model due to its effectiveness

in watermark identifiability and generation quality. The implementation of our experiments is based on fairseq [146]. We use the Adam optimizer [147] with $\beta = (0.9, 0.98)$ and set the learning rate to 0.0005. Additionally, we incorporate 4,000 warm-up steps. The learning rate then decreases proportionally to the inverse square root of the step number. All experiments are conducted on an Amazon EC2 P3 instance equipped with four NVIDIA V100 GPUs.

5.4.1 Main Results

The results of the watermark identifiability and generation quality for the text generation tasks studied in this chapter are presented in Table 5.1. We also show examples of watermarked text in Table 5.4. Both our method GINSEW and [117] achieve a 1.00 mean average precision (mAP) on the watermark detection, indicating the effectiveness of detecting IP infringements. Nevertheless, GINSEW has better BLEU and ROUGE-L scores, reflecting the better generation quality of our method. Note that when mAP reaches 1.00, the false positive rates become 0. This implies that by selecting an appropriate threshold (empirically set as $P_{\text{snr}} > 5.0$), the signal of unwatermarked models does not exceed this threshold. While [117] has perfect detection, we argue that the watermarks of extracted models in [117] can be easily erased as the synonym replacement techniques are not invisible. It's worth mentioning that GINSEW sees a negligible degradation in metrics such as BLEU, ROUGE, and BERTScore, when compared to the non-watermarked baseline. CATER utilizes conditional watermarks, which inevitably leads to non-obvious watermark signals in the extracted model's output.

5.4.2 Watermark removal attacks

To evaluate the effectiveness and robustness of our proposed method, we conduct a synonym randomization attack, also known as a watermark removal attack. Both [117] and CATER [131] use synonym replacement as a basis for their watermarking methods. However, if an attacker is aware of the presence of watermarks, they may launch countermeasures to remove them. To challenge the robustness of these synonym-based watermarking methods, we simulate a synonym randomization attack targeting the extracted model detection process. We use WordNet [148] to find synonyms of a word and create a list of word sets. These word sets may include words that are not semantically equivalent, so we use a pre-trained Word2Vec [149] model to filter out sets with dissimilar words. Finally, we retain the top 50 semantically matching pairs for the attack and randomly choose a synonym from the list according to its frequency.

In such a watermark removal attack, [117] and CATER fail to detect the adversary whereas GINSEW can perform much better in terms of mAP. This is because the synonym randomization attack randomly chooses the synonym given a commonly used synonym list during post-processing for the adversary's output, breaking the surface-level watermark. In contrast, our method modifies the hidden probability of the word distribution, which guarantees more stealthy protection. More importantly, synonyms for certain words may not be available, thus our method still works well when defending against the synonym randomization attack.

5.4.3 Case study

We conduct a case study as an example to demonstrate the watermarking mechanism in GINSEW. We use the IWSLT14 machine translation dataset to train the victim model, where we set the watermark level $\varepsilon = 0.2$ and the angular frequency $f_w = 16.0$ for the

victim model. To test the effectiveness of GINSEW, two different extracted models are created: a positive one and a negative one. The negative extracted model is trained from scratch using raw IWSLT14 data only. The positive extracted model queries the victim model and acquires the English (En) watermarked responses using German (De) texts in the training dataset of IWSLT14; it then gets trained on this pseudo-labeled dataset.

In the watermark detection process, we set the threshold $q_{\min} = 0.6$ and treat a subset of German inputs as the probing dataset. We use the watermark key K to extract the output of the extracted model following Algorithm 9, which enables us to analyze the group probability of positive and negative extracted models in both the time and frequency domains.

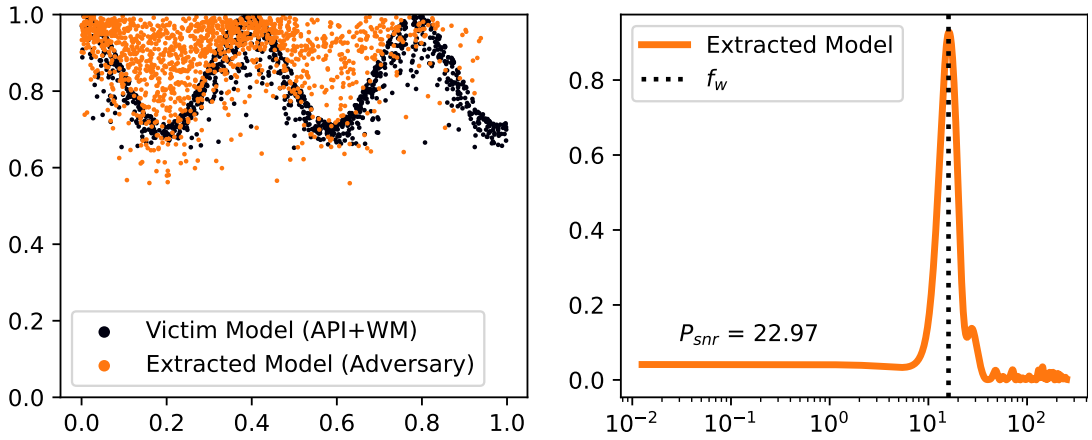

Figure 5.3: A positive example of GINSEW. There is a significant peak in the power spectrum at frequency f_w.

As shown in Figure 5.3, the output of the watermarked victim model follows an almost perfect sinusoidal function and that of the positive extracted model has a similar trend in the time domain. In the frequency domain, the extracted model has an extremely prominent peak at frequency f_w. The P_{snr} score exceeds 20 for the positive example, indicating a high level of watermark detection. Furthermore, Figure 5.4 illustrates that without the correct watermark key, the adversary is unable to discern whether the victim model API has embedded a watermark signal or not, as the hash function can be completely

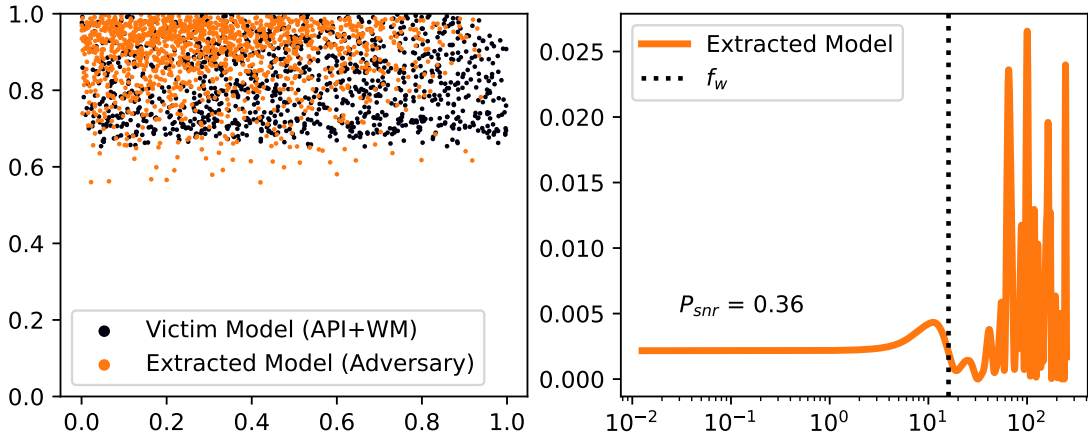

Figure 5.4: Use a wrong key to build the hash function for the positive example.

random. In this sense, GINSEW achieves stealthy protection of the victim model.

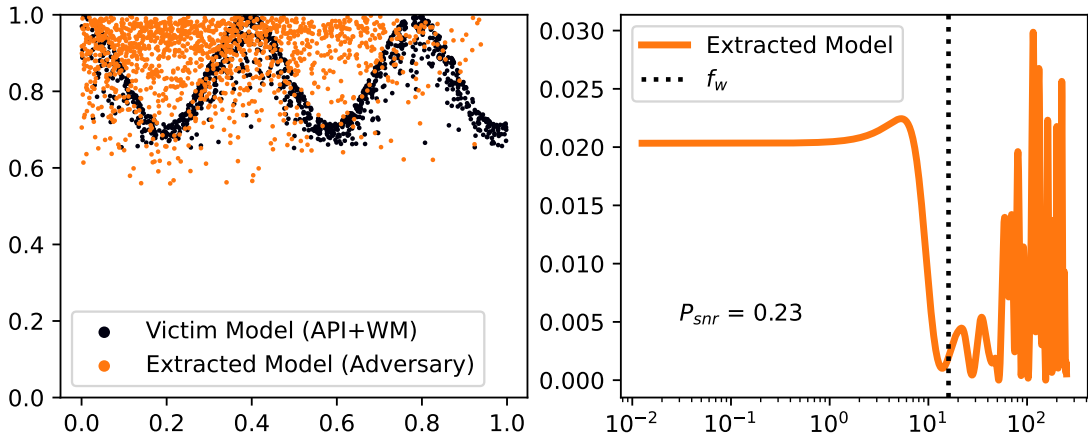

Figure 5.5: A negative example of GINSEW. There is no peak in the frequency domain.

In Figure 5.5, we show the negative example of a suspect model trained from scratch (without watermarked data). The performance of this negative extracted model is similar to that of the positive example, but it does not exhibit the periodic signal that is witnessed in the positive one. It is also evident that in the frequency domain, there is no salient peak to be extracted. Moreover, the P_{snr} score for the negative example is close to 0, which echoes the lack of a clear signal.

Ablation Study

5.4.4 Watermark detection with different architectures

	IWSLT14		ROCStories	
	BLEU	mAP	ROUGE-L	mAP
(m)BART	34.0	100	16.0	100
Transformer 6-6	34.2	100	16.1	100
Transformer 4-4	33.9	100	16.0	100
Transformer 2-2	33.2	64.4	15.5	53.7
ConvS2S	33.7	84.2	15.8	92.1

Table 5.2: Watermark detection with different model architectures. We choose four different architectures for the extracted model and report the generation quality and detection performance (mAP $\times$100).

GINSEW is designed to be independent of the model architectures, in other words, it can effectively protect against model extraction attacks regardless of the architectures of the extracted model. To demonstrate this, we conduct experiments using different model architectures for the extracted models. We report the results in Table 5.2. Adversaries are often unaware of the architecture of remote APIs, but recent studies have shown that model extraction attacks can still be successful when there is a mismatch between the architecture of the victim model and that of the extracted model (e.g., [101, 134]). Consequently, to test in this setting, we impose different architectures for the adversary models in the experiments. We use Transformer models with varying numbers of encoder and decoder layers, as well as a ConvS2S [150] model as the adversaries. Additionally, considering real-world scenarios where adversaries may start with pre-trained models to distill the victim model, we also include pre-trained models such as BART [52] and mBART [151] as adversaries. The adversaries are trained/fine-tuned based on the

The experiment results, summarized in Table 5.2, indicate that (m)BART and Transformer models with either 6 encoder-decoder layers or 4 encoder-decoder layers achieve

perfect detection performance for both datasets; the 2 encoder-decoder Transformer model and the ConvS2S model show worse detection performance. We conjecture that this is due to the fact that the latter models have fewer parameters, which makes it difficult for them to learn the hidden signal in the output of the victim model. Overall, the results suggest that our approach can effectively protect against model extraction attacks, regardless of the architecture of the extracted model.

5.4.5 Watermark detection with text alone

In Algorithm 9, we introduce a method for detecting watermarks using text probabilities $(t^{(k)}, \hat{Q}_{\mathcal{G}_1}^{(k)}) \in \widetilde{\mathcal{H}}$. Here, we explore the possibility of detecting watermarks by analyzing just the generated text itself. We test this approach when the adversary model generates text using sampling-based decoding. We collect pairs $(t^{(k)}, \mathbf{1}(\boldsymbol{y}_i^{(k)} \in \mathcal{G}_1))$ representing whether a given token belongs to group 1, $\mathcal{G}_1$, or not. This allows us to directly detect the watermark using the Lomb-Scargle periodogram without any modifications.

To conduct our experiments on IWSLT14 datasets, we employ the same adversary model (transformer 6-6) as shown in Table 5.1 and probe it to obtain text outputs. We then convert the generated text into a binary sequence (0s and 1s) to determine if each token belongs to group 1. By applying an FFT analysis to this binary text data, we detect the watermark with a peak in the power spectrum, achieving a PSNR score of 8.35. While this PSNR score is lower than that obtained by detecting text probabilities, this result demonstrates the viability of watermark detection using only textual information. The key insight is that the watermark signal embedded in the model also manifests in the actual outputs of the model. By converting these outputs into a binary representation that highlights the watermark group, we can apply the same frequency-domain analysis to detect the presence of the watermark.

5.4.6 The impact of watermark level

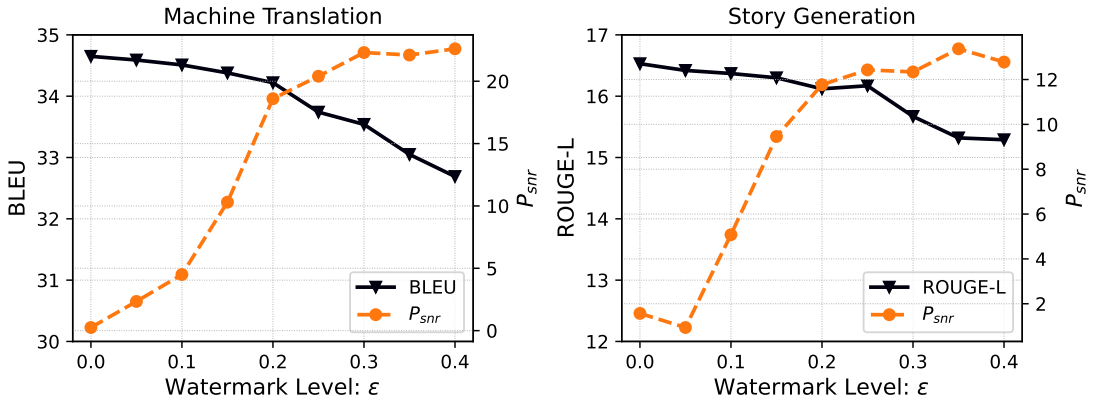

Figure 5.6: Generation quality for the victim model and detected P_{snr} score with different watermark levels on machine translation and story generation tasks.

The watermark level (ε) plays a crucial role in determining the effectiveness of the watermarking technique. The watermark level refers to the degree of perturbation added to the output of the victim model. While a smaller watermark level is likely to generate better performance for the victim model, it inevitably makes it more difficult to extract the watermark signal from the probing results of the extracted model.

As depicted in Figure 5.6, we observe that increasing the watermark level results in a decrease in generation quality on both machine translation and story generation tasks. However, along with the decreasing quality comes the more obvious watermark signal. When the watermark level is low ($\varepsilon < 0.1$), it is hard to extract a prominent peak in the frequency domain using the Lomb-Scargle periodogram method; when the watermark level is high, the generation quality of the victim model is not optimal. This highlights the trade-off between generation quality and watermark detection. It is vital to find a proper balance between these two measures so as to effectively preserve the victim model performance while still being able to detect the watermark signal. Taking this into account, in the experiments presented in Table 5.1 we select a watermark level of $\varepsilon = 0.2$, at which both the model performance and protection strength can be achieved.

5.4.7 Mixture of raw and watermarked data

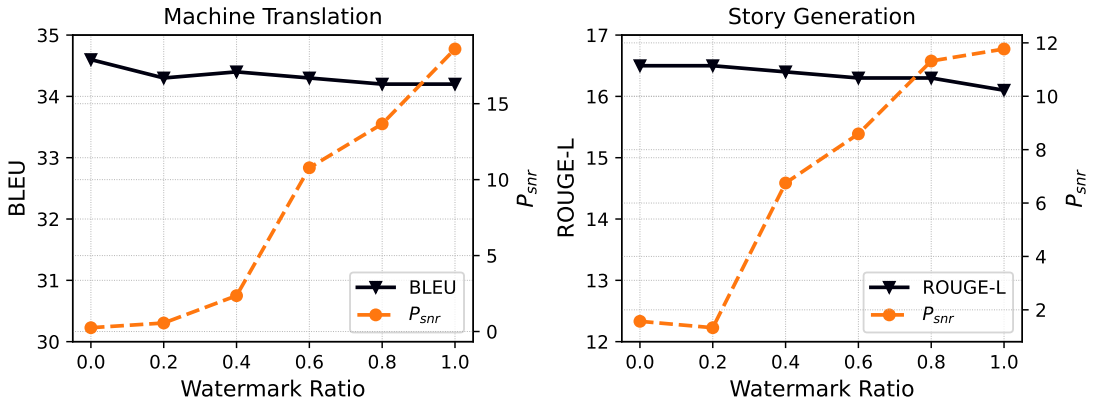

Figure 5.7: Impact of watermarked data ratio on generation quality and P_{snr} for the extracted model on machine translation and story generation tasks.

In the real world, adversaries often attempt to circumvent detection by the owner of the intellectual property (IP) they are infringing upon. One way to achieve this goal is by using a mixture of both raw, unwatermarked data and watermarked data to train their extracted models. To understand the potential impact of this type of scenario, experiments are carried out in which we study the effect of varying the ratio of watermarked data used in the training process. The experiment results in Figure 5.7 reveal that the extracted signal P_{snr} increases as the ratio of watermarked data rises. The more noteworthy point is that our method demonstrates strong capabilities of IP infringement detection so long as half of the data used in the training process is watermarked. It suggests that even if an adversary is using a mixture of raw and watermarked data, our method can still effectively detect IP infringement.

5.4.8 Different decoding methods

Our proposed method modifies the probability vector by embedding a watermark signal. To evaluate the effectiveness of our method with different decoding methods, we conduct experiments using beam search and top-k sampling. Specifically, we test

	IWSLT14		ROCStories	
	BLEU	P_{snr}	ROUGE-L	P_{snr}
Beam-5	34.2	18.3	16.1	11.4
Beam-4	34.1	19.4	16.1	12.2
Top-5 Sampling	31.5	23.3	13.4	13.8

Table 5.3: Watermark detection with different decoding methods. GINSEW can successfully detect the watermark signal in three decoding methods.

beam search with a size of 5 and 4, as well as top-k sampling with k=5. We measure both the generation quality and the strength of the watermark signal, the results of which are displayed in Table 5.3. Beam search yields better generation quality, and as a consequence, we use beam search with a size of 5 as the default decoding method in our experiments in Table 5.1. Nonetheless, results in Table 5.3 validate that our method, in general, remains robust and effective when different decoding methods are employed. For all the three decoding methods tested, our method can successfully detect a prominent signal in the frequency domain, further corroborating the robustness of GINSEW.

5.4.9 How does watermark signal change in the distillation process?

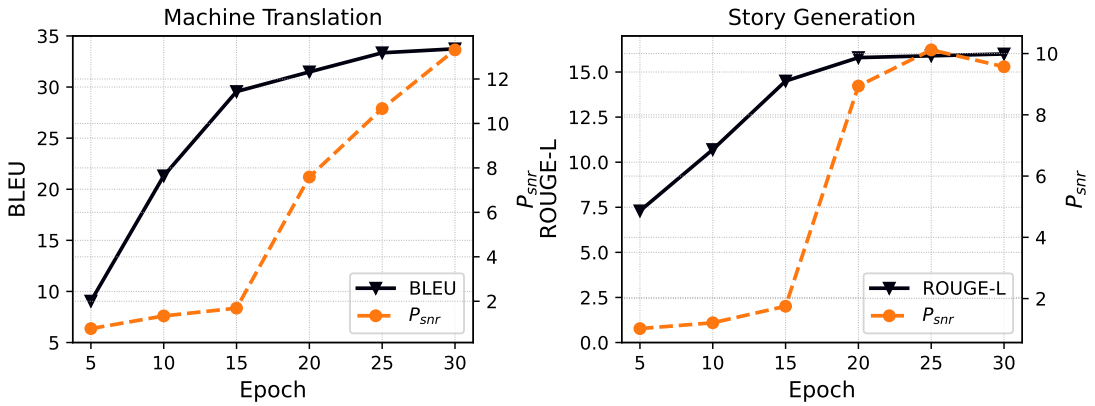

Figure 5.8: Performance and detected P_{snr} score of the extracted model with different epochs on machine translation and story generation tasks.

To study the impact of the distillation process on the watermark signal, we conduct an experiment to explore how the generation quality and P_{snr} scores change with the number of training epochs. As shown in Figure 5.8, the quality of the generated text improves as the number of training epochs increases. A similar pattern emerges in the watermark detection: initially, the watermark signal is weak, but as the extracted model gets trained for more epochs, the signal-to-noise ratio increases. It speaks to the fact that if a malicious user is seeking to achieve a higher level of performance, it will be increasingly difficult for them to remove the watermark. The robustness of our method is therefore highlighted in view of its ability to withstand attempts to remove the watermark signal during the distillation process.

5.5 Conclusion

In this chapter, we propose GINSEW to generate an invisible sequence watermark for protecting language generation models from model extraction attacks. Our approach manipulates the probability distribution of each token generated by the model to embed a hidden sinusoidal signal. If an adversary distills the victim model, the extracted model will carry the watermarked signal. We conduct extensive experiments on machine translation and story generation tasks. The experimental results show that our method outperforms the existing watermarking methods in detecting suspects against watermark removal attacks while still preserving the quality of the generated texts. Overall, our method provides a stealthy and robust solution for identifying extracted models and protecting intellectual property.

Limitations. As shown in Section 5.4.6 and 5.4.7, the effectiveness of GINSEW is limited when the watermark level is low or only a small portion of the training data is

watermarked for the adversary. Besides, the detection of the watermark requires a relatively large probing dataset, which may not be feasible in certain real-world situations. Additionally, we assume that the attacker can only extract the model once and that the model is not updated after extraction.

Additional Results and Proofs

5.5.1 Watermarked examples

Example 1: Unwatermarked: first of all, because the successes of the Marshall Plan have been overstated. Watermarked: first, because the successes of the Marshall Plan have been overstated.
Example 2: Unwatermarked: because life is not about things Watermarked: because life isn't about things
Example 3: Unwatermarked: i was at these meetings i was supposed to go to Watermarked: i was at the meetings i was supposed to go to

Table 5.4: Watermarked examples

5.5.2 Distribution property

Lemma 5.5.1 (Lemma 1 in [8]). *Assume* $\mathbf{v} \sim \mathcal{U}(0,1),\ \mathbf{v} \in \mathbb{R}^n$ *and* $\mathbf{x} \sim \mathcal{N}(0,1),\ \mathbf{x} \in \mathbb{R}^n$, *where* $\mathbf{v}$ *and* $\mathbf{x}$ *are both i.i.d. and independent of each other. Then we have:*

$$\frac{1}{\sqrt{n}}\mathbf{v}\cdot\mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right),\ n \to \infty$$

Proof. Let $u_i = \mathbf{v}_i\mathbf{x}_i,\ i \in 1, 2, \ldots, n$. By assumption, u_i are *i.i.d.*. Clearly, the first and second moments are bounded, so the claim follows from the classical central limit

theorem,

$$\sqrt{n}\bar{u}_n = \frac{\sum_{i=1}^{n} u_i}{\sqrt{n}} \rightsquigarrow \mathcal{N}\left(\mu, \sigma^2\right) \quad \text{as} \quad n \to \infty$$

where

$$\begin{aligned}
\mu &= \mathbb{E}\left(u_i\right) = \mathbb{E}\left(\mathbf{v}_i \mathbf{x}_i\right) = \mathbb{E}\left(\mathbf{v}_i\right) \mathbb{E}\left(\mathbf{x}_i\right) \\
&= 0 \\
\sigma^2 &= \operatorname{Var}(u_i) = \mathbb{E}\left(u_i^2\right) - \left(\mathbb{E}\left(u_i\right)\right)^2 \\
&= \mathbb{E}\left(u_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2 \mathbf{x}_i^2\right) = \mathbb{E}\left(\mathbf{v}_i^2\right) \mathbb{E}\left(\mathbf{x}_i^2\right) \\
&= \frac{1}{3}
\end{aligned}$$

It follows that given large n

$$\frac{1}{\sqrt{n}} \mathbf{v} \cdot \mathbf{x} \rightsquigarrow \mathcal{N}\left(0, \frac{1}{3}\right)$$

□

5.5.3 Modified group probability properties

As we discussed in Section 3.2, the watermarked distribution produced by our method remains a valid probability distribution. The following lemma formally establishes this result.

Lemma 5.5.2. *Let $Q_{\mathcal{G}_1}$ and $Q_{\mathcal{G}_2}$ be the group probability in probability vector $\mathbf{p}$, then the modified group probability, as defined in Equation 5.3, 5.4 satisfies $0 \leq \tilde{Q}_{\mathcal{G}_1}, \tilde{Q}_{\mathcal{G}_2} \leq 1$ and $\tilde{Q}_{\mathcal{G}_1} + \tilde{Q}_{\mathcal{G}_2} = 1$.*

Proof. Notice that $Q_{\mathcal{G}_1}$ and $Q_{\mathcal{G}_2}$ are the summation of the token probabilities in each group, we have

$$0 \leq Q_{\mathcal{G}_1}, Q_{\mathcal{G}_2} \leq 1 \text{ and } Q_{\mathcal{G}_1} + Q_{\mathcal{G}_2} = 1.$$

Given $z_1(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M})\right)$ and $z_2(\boldsymbol{x}) = \cos\left(f_w g(\boldsymbol{x}, \mathbf{v}, \mathbf{M}) + \pi\right)$, we have

$$0 \leq z_1(\boldsymbol{x}), z_2(\boldsymbol{x}) \leq 1 \text{ and } z_1(\boldsymbol{x}) + z_2(\boldsymbol{x}) = 0$$

Then we can get

$$0 \leq Q_{\mathcal{G}_1} + \varepsilon\left(1 + z_1(\boldsymbol{x})\right) \leq 1 + 2\varepsilon$$
$$0 \leq Q_{\mathcal{G}_2} + \varepsilon\left(1 + z_2(\boldsymbol{x})\right) \leq 1 + 2\varepsilon$$

Therefore,

$$0 \leq \frac{Q_{\mathcal{G}_1} + \varepsilon\left(1 + z_1(\boldsymbol{x})\right)}{1 + 2\varepsilon} \leq 1$$
$$0 \leq \frac{Q_{\mathcal{G}_2} + \varepsilon\left(1 + z_2(\boldsymbol{x})\right)}{1 + 2\varepsilon} \leq 1$$

$$\begin{aligned}\frac{Q_{\mathcal{G}_1} + \varepsilon\left(1 + z_1(\boldsymbol{x})\right)}{1 + 2\varepsilon} + \frac{Q_{\mathcal{G}_2} + \varepsilon\left(1 + z_2(\boldsymbol{x})\right)}{1 + 2\varepsilon} &= \frac{\left(Q_{\mathcal{G}_1} + Q_{\mathcal{G}_2}\right) + 2\varepsilon + \varepsilon\left(z_1(\boldsymbol{x}) + z_2(\boldsymbol{x})\right)}{1 + 2\varepsilon} \\ &= \frac{1 + 2\varepsilon}{1 + 2\varepsilon} = 1\end{aligned}$$

□

Part III

Privacy-Preserving LLMs

Chapter 6

Provably Confidential Language Modelling

Large language models are shown to memorize privacy information such as social security numbers in training data. Given the sheer scale of the training corpus, it is challenging to screen and filter all privacy data, either manually or automatically. In this chapter, we propose **C**onfidentially **R**edacted **T**raining (CRT), a method to train language generation models while protecting the confidential segments. We borrow ideas from differential privacy (which solves a related but distinct problem) and show that our method is able to *provably prevent* unintended memorization by randomizing parts of the training process. Moreover, we show that redaction with an approximately correct screening policy *amplifies* the confidentiality guarantee. We implement the method for both LSTM and GPT language models. Our experimental results show that the models trained by CRT obtain almost the same perplexity while preserving strong confidentiality. Our code is available at https://github.com/XuandongZhao/CRT.

6.1 Introduction

Language models (LM) have rich real-world applications in, among others, machine translation [152], AI chatbots [153], question answering [154], and information retrieval [155]. The advent of transformers [156] has fostered a dramatic advancement in the capabilities of generative neural language models, yet they come at a cost to privacy, as the amount of excess parameters in the LM enables it to memorize certain training samples. Recent works show that sensitive user information from the training dataset, such as address and name, can be extracted verbatim from text generation models by querying the LM as an API [157, 158, 159]. How to train a high-performing language model without memorizing sensitive text has become a major research challenge.

Existing solutions to this problem primarily leverage differential privacy (DP) [43].

Differentially private learning algorithms ensure that an attacker could not infer whether a data point is used for training, let alone extracting the sensitive information within that data point.

However, there are several mismatches between the problem of *privacy* that DP addresses, and our problem of preventing the memorization of sensitive text (henceforth referred to as *confidentiality*). First, confidential information in a natural language dataset is sparse (e.g., the bulk of an email might not carry confidential information). DP's undiscriminating protection for all sentences could be unnecessarily conservative which limits the utility of the trained model. Second, what needs to be protected is the content of the sensitive text, rather than the data context. For example, in the sentence ‘‘`My SSN is 123-45-6789.`’’, it is the actual SSN that we hope to conceal rather than the general information that someone entered her SSN in a chatbot dialogue. Thirdly, the same sensitive content could appear in many data points, which makes the protection of the content more challenging than protecting one data sample. These differences mo-

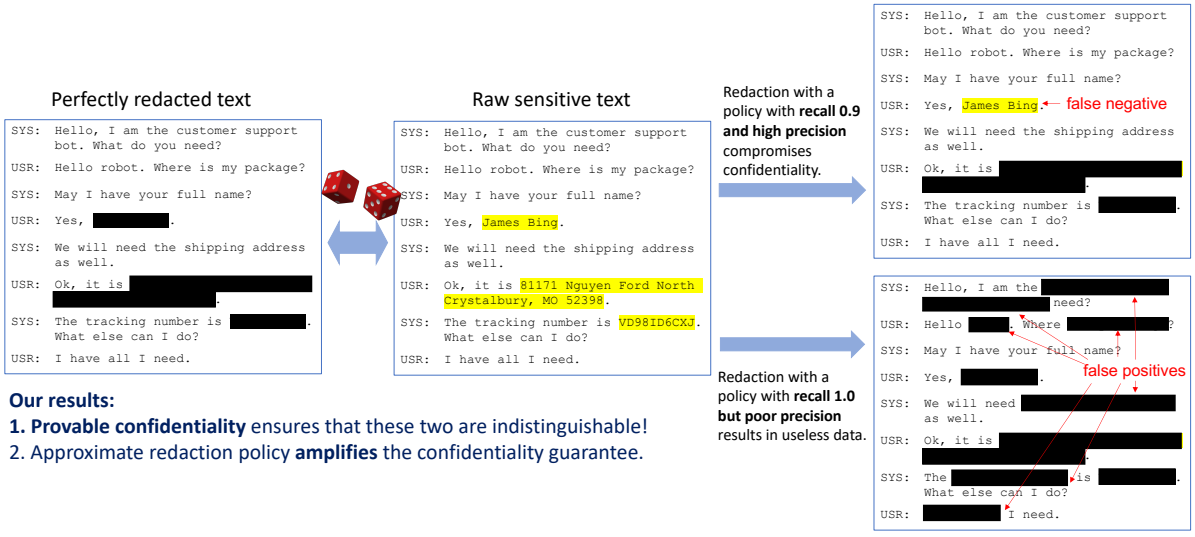

Figure 6.1: An example from simulated dialog dataset `CustomerSim`. The yellow highlights are confidential content (middle). Left shows the text after *Redaction* by a sequence labeling policy π. However, if the policy is not perfect, there exists false negative or false positive samples as shown on the right.

tivate us to treat the problem of confidentiality protection in LM separately with new definitions.

Besides DP, we also consider classical techniques of redaction and deduplication. *Redaction* refers to the process of removing sensitive or classified information from a document prior to its publication in governmental and legal contexts. *Deduplication* is the procedure of detecting and removing identical and nearly identical texts from a corpus. The main challenge of applying these techniques is that it is hard to manually redact a gigantic dataset and automated tools are far from being perfect.

The contribution of this chapter is fivefold.

1. We show that in the absence of a perfect screening policy, the risk of a language model memorizing sensitive content is real and can be efficiently exploited with only blackbox access to the model even if the learning algorithm satisfies the recently proposed notion of *selective differential privacy* [160].

2. Inspired by differential privacy, we introduce a new definition of *confidentiality* which precisely quantifies the risk of leaking sensitive text.

3. We propose CRT to train language generation models while protecting confidential text. The method with deduplication and redaction operations work even under imperfect confidential text labeling policies.

4. We theoretically prove that CRT, combined with differentially private stochastic gradient descent (DP-SGD), provides strong confidentiality guarantees.

5. Our experiments on both MultiWOZ 2.2 and CustomerSim datasets show that different models trained by CRT can achieve the same or better perplexity than existing solutions (against the attacks of [157, 158]).

To the best of our knowledge, we are the first to rigorously establish the role of deduplication and redaction in achieving provably stronger confidentiality (or the related differential privacy) guarantees and the first to achieve provably confidentiality in transformer models with only a mild utility loss.

6.2 Related Work

Next, we briefly introduce the relevant background and discuss the related work to put our work in context.

Language modeling is a fundamental problem in natural language processing [161, 162, 163]. Consider a text sequence that consists of multiple tokens from a vocabulary $\mathcal{V}$, i.e., $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, where w_i is the i-th token. The goal of language modeling is to construct a generative model of the distribution $\Pr(\boldsymbol{w})$, by applying the chain rule $\Pr(\boldsymbol{w}) = \prod_{i=1}^{n} \Pr\left(w_i \mid \boldsymbol{w}_{<i}\right)$. We let $f_\theta(w_i|\boldsymbol{w}_{<i})$ denote the likelihood

of token w_i when evaluating the neural network f with parameters θ. A language model is trained to maximize the probability of the data in a training set $\mathcal{W}$, by minimizing the negative log-likelihood over each training example with the loss function $\mathcal{L}(\theta) = -\log \prod_{i=1}^{n} f_\theta\left(w_i \mid \boldsymbol{w}_{<i}\right)$. Recurrent neural networks (RNNs) used to be a common choice for the neural network architecture to estimate the probability distribution $\Pr(\boldsymbol{w})$. [164, 165]. More recently, large-scale Transformer-based language models have replaced RNNs in state-of-the-art models for all sorts of NLP tasks [156, 40]. Nevertheless, common language models are vulnerable to privacy attacks and possibly expose information about their sensitive training data [157, 158].

Differentially private (DP) learning methods [166, see, e.g.,] have been applied to language models as a blanket solution for a number of privacy and security risks. [167] trained an RNN language model with DP guarantees in a federated learning setup. [168] pre-trained BERT under DP on datasets with hundreds of millions of examples. These papers also demonstrated that DP can effectively prevent data-extraction attacks in practice even for algorithms with DP guarantees that are considered too weak from a theoretical-perspective (e.g., $\epsilon = 8$ or 16). However, the strong protection of DP often results in a substantial drop in the utility of the trained model, which makes them less desirable in practice. In fact, it was recently shown that it is *necessary* for deep learning models to memorize certain training data to achieve high accuracy [169], which suggests that DP or any other techniques that require the model to not memorize any training data will perform poorly in the high-dimensional, power-law distributed real datasets. This motivates us to consider weakened models that only prevent memorizing the sensitive part of the text.

Recent works [159, 170] show that deduplication enables language models to emit memorized text less frequently with the same or better accuracy. However, deduplicating training datasets can not prevent all unintended memorization. We combine dedu-

plication and redaction and then apply both techniques to the training process of LM to achieve confidentiality with a provable guarantee.

The closest to us is perhaps the work of [160], who proposed *selective differential privacy* (S-DP), which requires indistinguishability between two datasets that differ only on a sensitive message. Correspondingly, they propose an algorithm (Selective DP-SGD) for training RNN that adds noise only to the part of computation that involves sensitive tokens. To define S-DP and to run Selective DP-SGD, one needs to have access to a policy function F which determines which token is sensitive. This requirement limits the applicability of their approach to those applications where such perfect F is known. We note that even for name-entity recognition the state-of-the-art model is far from being perfect, and which part of the text is considered sensitive is often ambiguous even for human annotators. We will see that naively running Selective DP-SGD with an approximate policy function does not provide a meaningful confidentiality guarantee and is vulnerable to practical data extraction attacks. Finally, we note that in the case when a perfect policy function is available, we can simply use it for redaction, which provides a perfect S-DP with $\epsilon = 0$. A big part of our contribution is to refine S-DP to a (slightly different) definition called "confidentiality" and to demonstrate that we use an approximate screening policy to amplify the confidentiality parameter.

6.3 Proposed Method: CRT

In this section, we develop our method with provable confidentiality.

6.3.1 Formally defining confidentiality

Let the dataset be a collection of n data points — each being a sequence of tokens. A "secret" x is a contiguous subsequence of tokens within a data point that is considered

sensitive or *confidential*. The goal of our research is to allow us to train language models on such datasets that could contain secrets while provably prevent the model from remembering that these secrets were. We start by defining a formal definition of confidentiality, which uses the following idea of indistinguishability from the DP literature.

Definition 6.3.1 (Indistinguishability)**.** We say that a pair of distributions P, Q defined on the same probability space are (ϵ, δ)-indistinguishable if for any measurable set S,

$$\Pr_P[S] \leq e^\epsilon \Pr_Q[S] + \delta.$$

Definition 6.3.2 (Confidentiality)**.** We say that $\mathcal{A}$ ensures that a secret x is $(\epsilon(x), \delta)$-confidential, if for any dataset D that contains x in one of its data points, and an alternative dataset D' that replaces x in D with a generic `<MASK>`, it holds that $(\mathcal{A}(D), \mathcal{A}(D'))$ are $(\epsilon(x), \delta)$-indistinguishable. In addition, we simply say that $\mathcal{A}$ ensures (ϵ, δ)-confidentiality if $\epsilon(x) \leq \epsilon$ for all secret x.

This definition ensures that an attacker cannot distinguish from the output of $\mathcal{A}$ (the trained language model) whether it was x or `<MASK>` that was used for training, thus formalizing the idea of confidentiality. The protection should be viewed as relative, rather than absolute. The definition bounds the risk of any bad event by an multiplicative factor of e^ϵ and an additive factor of δ, which implies that anything that could happen when we run $\mathcal{A}$ on the sensitive data could've happened with with similar probability even if $\mathcal{A}$ runs on an alternative world where these sensitive information are perfectly masked.

Connections to differential privacy. Our definition of confidentiality is related to (and inspired by) (ϵ, δ)-differential privacy (DP) but is different in several ways. DP is stronger (and implies confidentiality!) requires $\mathcal{A}$ to ensure (ϵ, δ)-indistinguishability

for all D, D' that can be modified from each other by adding or removing one individual person / data point (or tokens, depending on the desired granularity); but for $\mathcal{A}$ to ensure (ϵ, δ)-confidentiality, it only requires (ϵ, δ)-indistinguishability for specific D, D' where D' replaces x in D with `<MASK>`. Moreover, it is more informative to define ϵ as a function of each specific x, which is different from DP (it resembles personalized DP [171]).

The confidentiality definition makes sense for our problem because it protects the content of the sensitive text x rather than its existence. Specifically, a pre-processing algorithm that masks all sensitive text ensures $(0, 0)$-confidentiality but does not satisfy any non-trivial DP guarantees.

Sometimes, it is useful to consider the confidentiality of multiple secret texts. For example, a secret key x could appear multiple times in multiple data points. Also, there might be multiple secret texts that are correlated to each other such that the knowledge of one would reveal other secrets.

Definition 6.3.3 (Group Confidentiality)**.** We say that $\mathcal{A}$ ensures that a **list** of sensitive texts $\mathcal{S} := [x_1, ..., x_k]$ is $(\epsilon(\mathcal{S}), \delta)$-(group) confidential, if for any dataset D that contains $[x_1, ..., x_k]$ in up to k data points, and D' being the version that replaces each element in $\mathcal{S}$ with `<MASK>`, it holds that $(\mathcal{A}(D), \mathcal{A}(D'))$ are $(\epsilon(\mathcal{S}), \delta)$-indistinguishable.

A special case of such group confidentiality is when $\mathcal{S}$ collects the *all secret text* in D, which protects all secret texts uniformly. We call this *uniform-confidentiality*. Note that the standard definition of confidentiality also protect every secret x, except that it protects each secret x individually, rather than together.

Inspired by the recent development of Bayesian DP [172], we also define Bayesian confidentiality as follows.

Definition 6.3.4 (Bayesian Confidentiality)**.** Let D be a dataset that is fixed except a random secret $x \sim \mu$ drawn from some distribution μ. Let D' be obtained by re-

placing x with <MASK>[1]. Then $\mathcal{A}$ ensures (ϵ, δ)-Bayesian Confidentiality if for any D', $(\mathcal{A}(D), \mathcal{A}(D'))$ is (ϵ, δ)-indistinguishable, where $\mathcal{A}(D)$ is jointly distributed over $x \sim \mu$ and $\mathcal{A}$.

The Bayesian confidentiality measures how much information an attacker could gain if he/she's prior knowledge about this secret x is described by the distribution μ. This is a strict generalization because when μ is a single point mass at x, it recovers Definition 6.3.2. The additional generality allows us to quantify the stronger confidentiality guarantee against weaker adversaries without complete information.

6.3.2 Confidentially redacted training

In this section we describe the CRT method to train language models with provable confidentiality guarantee. It includes two pre-processing operations (deduplication and redaction) and a switching optimization procedure. The overall idea is to screen the corpus into two separate sets, one public set including sentences with no confidential information, and one private set including sentences containing confidential content. We then use normal optimization algorithms (e.g. SGD) on the public set and differential privacy optimizer (e.g. DP-SGD) on the private set.

Deduplication. The deduplication procedure Dedup detects all sentences that appear multiple times in the training data and replace them into a single <MASK> from the second occurrence onwards (<MASK> is for proving purpose).

Redaction. The redaction procedure Redact_π takes applies a sequence labelling policy π to screen confidential content in the training corpus D. $\pi(s, x) = 1$ if a token x in a sentence s should be confidential. The labeled span in each detected sentence is replaced

[1]Notice that D' is fixed even though x is random.

with a special token <MASK>. Note that we do not assume the policy is perfect. It may label some non-sensitive tokens as sensitive (false positives) and label some sensitive text as non-sensitive (false negative, or 1−recall).

Redact and Dedup could be implemented manually, but with the large text corpus nowadays it is more common that these procedures are implemented using automated tools. For example, Dedup could be implemented efficiently with just one pass of data using a *bloom filter* [173] (or other hashing tricks that also catches near-duplicates). Bloom filter in particular, enjoys the nice property that it could have false positives but never any false negatives. Redact_π could be realized by a named entity recognition (NER) model or a personal-identifiable information (PII) detector.

Algorithm 11 Confidentially Redacted Training (CRT)

Input: Dataset D (after tokenization / splitting), labelling policies π, π_c, number of epochs T
$D' \leftarrow \text{Dedup}(D)$
$D'' \leftarrow \text{Redact}_\pi(D')$
$D^{pri} \leftarrow \{s \in D'' \mid \exists x \in s \text{ s.t. } \pi(s,x) = 1 \text{ or } \exists x \subset s \text{ s.t. } \pi_c(s,x) = 1\}$
$D^{pub} \leftarrow \{s \in D'' \mid s \notin D^{pri}\}$
for $e = 1$ **to** T **do**
    Run one epoch of SGD with D^{pub}
    Run one epoch[2] of DP-SGD with D^{pri}
end for

Finally, CRT combines the two pre-processing steps with normal optimizer and DP-SGD, the standard algorithm for deep learning with differential privacy. A pseudo-code of the algorithm is given in Algorithm 11.

Besides using a sequence labeling policy π with balanced precision/recall as part of the redaction process. The algorithm uses another, more conservative, policy π_c with nearly perfect recall to decide on the data points that do not contain sensitive text. In

[2]DP-SGD uses Poisson-sampled Gaussian mechanisms (with a random batchsize), thus cannot ensure all data points are seen and some data points might be seen many times. One epoch means the number of iterations that in expectation covers $|D^{pri}|$ data points.

the situation when such π_c isn't available, we simply choose $\pi_c(s, x) = 1$ for all tokens x in a sentence s and the second part becomes the vanila DP-SGD. It is also important that every data point that contains a `<MASK>` requires protection.

6.3.3 Theoretical analysis

We analyze the theoretical properties of the above method and show that they result in provable improvements in the (regular, group and Bayesian) confidentiality parameters for any algorithms that are provably $(\epsilon(x), \delta)$-confidential as defined in Section 6.3.1.

The following theorem captures the benefit of redaction in improving confidentiality.

Proposition 6.3.5 (Confidentiality under redaction)**.** *If $\mathcal{A}$ ensures $(\epsilon(x), \delta)$-Confidentiality for each token x of sentence $s \in \mathcal{S}$ ($\mathcal{S}$ is a corpus), then $\mathcal{A} \circ \text{Redact}_\pi$ ensures $(\tilde{\epsilon}(x), \delta)$-confidentiality with*

$$\tilde{\epsilon}(x) = \begin{cases} \epsilon(x) & \text{if } \pi(s, x) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

In addition, $\mathcal{A} \circ \text{Redact}_\pi$ also satisfies $(\tilde{\epsilon}(S), \tilde{\delta}(S))$-group confidentiality with

$$\tilde{\epsilon}(S) = \sum_{x \in s \& s \in \mathcal{S}} \epsilon(x) \mathbf{1}(\pi(s, x) = 0),$$
$$\tilde{\delta}(S) = \tilde{k} e^{\tilde{\epsilon}(S)} \delta$$

where $\tilde{k} := \sum_{x \in S} \mathbf{1}(\pi(s, x) = 0)$.

As an application of the above, if $\mathcal{A}$ ensures (ϵ, δ)-confidentiality, and that the empirical recall rates of the redaction policy on D is $1 - \gamma$, then the above proposition suggests that $\mathcal{A} \circ \text{Redact}_\pi$ improves the uniform-confidentiality over applying $\mathcal{A}$ without redaction by a factor of γ. The proof is in the appendix.

Redaction also improves Bayesian confidentiality in a way that mirrors the privacy amplification by sampling from the DP literature.

Proposition 6.3.6 (Bayesian Confidentiality under Redaction)**.** *If $\mathcal{A}$ ensures (ϵ, δ)-Bayesian Confidentiality with respect to $\mu[x|\pi(s,x) = 0]$ for a token x in a sentence s, then $\mathcal{A} \circ \text{Redact}_\pi$ ensures $(\log(1 + \gamma(e^\epsilon - 1)), \gamma\delta)$-Bayesian Confidentiality under μ if π has a false negative rate (i.e., $1-$"Recall") of γ under μ.*

The proposition says that if the redaction policy is accurate for secrets $x \sim \mu$, then we can have a stronger confidentiality parameter that scales roughly at $\tilde{\epsilon} = O(\gamma\epsilon)$. The idea behind the proof is that over the distribution of $x \sim \mu$, with probability $1 - \gamma$, $\text{Redact}_\pi(D) = \text{Redact}_\pi(D')$, thus $\mathcal{A} \circ \text{Redact}_\pi(D) \equiv \mathcal{A} \circ \text{Redact}_\pi(D')$. With probability γ, $\text{Redact}_\pi(D), \text{Redact}_\pi(D')$ are different and conditioning on the fact that Redact_π fails to detect x. Note that π is also applied to other text that are not sensitive, and could result in false positives, but they do not matter as the modification of Redact_π to D and D' will be identical. A full proof is given in the appendix.

Next we turn to deduplication.

Proposition 6.3.7 (Group confidentiality under deduplication.)**.** *If $\mathcal{A}$ ensures $(\epsilon(S), \delta(S))$-Group Confidentiality, then $\mathcal{A} \circ \text{Dedup}$ ensures $(\epsilon(\text{Unique}(S)), \delta(\text{Unique}(S)))$-Group Confidentiality.*

Deduplication provides a stronger protection for those cases where some secret x could appear multiple times in the dataset.

Theorem 6.3.8. *Let DP-SGD from Algorithm 11 satisfies (ϵ, δ)-differential privacy.*

1. *Assume $\pi_c(s,x) = 1$ for all secret tokens x in a sentence s such that $\pi(s,x) = 0$, then Algorithm 11 satisfies $(\epsilon \mathbf{1}(\pi(s,x) = 0), \delta)$-confidentiality.*

2. *Let S be a group containing m unique secrets such that $\pi_c(s, x) = 1 \forall x \in s$ and $s \in \mathcal{S}$ and that π detects $\tilde{\gamma}$-proportion of the unique secrets in S. Then Algorithm 11 satisfies $(\tilde{\gamma}m\epsilon, \tilde{\gamma}me^{\tilde{\gamma}m\epsilon}\delta)$-group confidentiality for S.*

3. *Let π, π_c has a recall of $1-\gamma$ and $1-\delta_2$ respectively on μ, then Algorithm 11 satisfies $(\log(1 + \gamma(e^{\epsilon} - 1)), \gamma\delta + \delta_2)$-Bayesian Confidentiality for μ.*

The theorem demonstrates that our CRT algorithm enjoys a full suite of confidentiality guarantees and they all benefit from the deduplication and redaction, particularly if π has high recall.

Note that the CRT algorithm achieves the worst-case confidentiality guarantee if we have a nontrivial conservative screening policy that outputs $\pi_c(x) = 1$ for *all* secret x that π misses, or we simply run vanilla DP-SGD after deduplication and redaction. On the other hand, CRT still satisfies Bayesian confidentiality for each μ depending on the recall rate of π_c under μ.

6.4 Experiments

We evaluate CRT by training two types of language model, LSTM and GPT-2, on two datasets: 1) MultiWOZ 2.2, a well-known human-written dialogue dataset and 2) CustomerSim, a simulated dialogue dataset for conversation generation.

MultiWOZ 2.2 is an already-public dialogue dataset written by crowd-workers, which collects over 10,000 annotated dialogues spanning 8 domains [174]. We use this dataset to show how CRT works in real-world applications. Following US Department of Labor's guidance[3] on personal-identifiable information (PII), we treat all confidential information (e.g. email address, reference number, telephone number, etc.) as secrets. For the

[3]https://www.dol.gov/general/ppii

sequence labeling policy π and conservative policy π_c, we build upon an NER model to do redaction. See Appendix 6.5.4 for more details.

CustomerSim. Following S-DP [160], we simulate a dialog dataset called CustomerSim with synthetic user information. The dialog flow is simulated based on a fixed agenda and the language generation is template-based [175]. CustomerSim consists of 10 thousand examples and over one million tokens. We treat user name, address, phone number, order, and tracking number as secrets, and use a regular expression tester (regex) to detect them for the redaction process.

Experiment details. For LSTM model, we follow the setting in S-DP to choose a one-layer LSTM. Because S-DP requires hidden states of the sensitive input to be protected, it doesn't support more layers nor Bidirectional LSTM. Since the advent of Transformers [156] significantly improves the capabilities of generative language models, we also test transformer-based language model GPT-2 [40] from HuggingFace [50]. As for deduplication, we use `SHA-1` [176] hash function to encode sequences to `SHA-1` hash code and then remove identical sequences based on the same hash code. For Bayesian Confidentiality, we treat the uniform distribution over the secret sequences as the distribution μ. More experiment details can be found in Appendix 6.5.3.

Baselines. For LSTM model, we compare four different training approaches: (1) vanilla SGD (denoted by "Non-private-LSTM"), (2) Selective DPSGD (denoted by "S-DP-LSTM") (3) DPSGD (denoted by "DPSGD-LSTM") and (4) confidentially redacted training (denoted by "CRT-LSTM"). While for GPT-2 model, we compare three different training approaches: (1) vanilla SGD (denoted by "Non-private-GPT"), (2) DPSGD (denoted by "DPSGD-GPT") and (3) CRT (denoted by "CRT-GPT"). Our implemen-

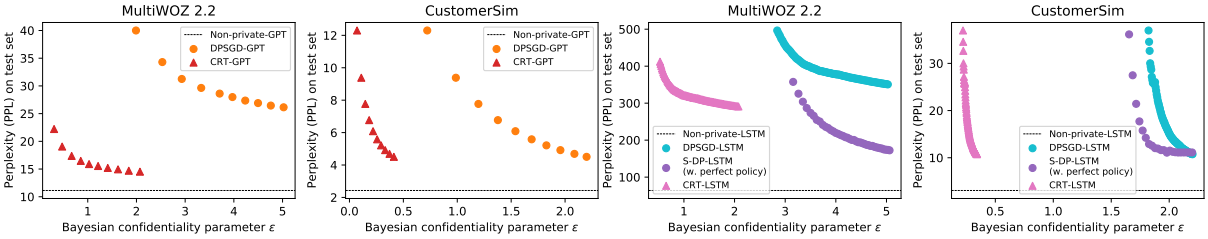

Figure 6.2: Model utility and confidentiality guarantee on MultiWOZ 2.2 and CustomerSim datasets with μ being a uniform distribution over the secret sequences in each dataset. PPL: Perplexity on the test set. ϵ: Privacy guarantee in Bayesian Confidentiality. We fix $\delta = 8e-5$ for all models. Since Selective DP-SGD with approximate policy gives $\epsilon = +\infty$, we show its result with a perfect screen policy. But when a perfect policy is available, `Redaction` only gives $\epsilon = 0$ and achieves the PPL of vanilla training with no noise added (Non-private-GPT/LSTM). For other models we set $\gamma = 0.1$ to show the result under approximate policy.

tation of S-DP-LSTM model is built upon [160][4]. We run the experiment for the GPT-2 model following [177][5], in which they propose ghost clipping method to alleviate the computational challenge of running DP-SGD with large Transformers.

All the models are trained five times to reduce randomness, and the parameters are tuned based on the validation set performances.

Experimental Results

6.4.1 Evaluation procedure

We need to evaluate both model utilities and privacy guarantees of the language models. We measure predictive perplexity (PPL) for the quality of LM. We also analyze the theoretical privacy budget (ϵ, δ) and test whether language models are private under attacks detailed below.

[4]https://github.com/wyshi/lm_privacy
[5]https://github.com/lxuechen/private-transformers

Canary insertion attack. Canary insertion is proposed as a testing methodology for quantitatively assessing the risk of *unintended memorization* [157]. It inserts random sequences called canaries into the training dataset, then trains the model, and finally calculates the following exposure for the inserted canaries to measure a model's potential for privacy risks. In our experiment, we randomly generate 10 canaries in the form of "`My ID is: <random 6-digit number here>`". Each canary is inserted into the training dataset 20 times to generate more salient differences between models.

Definition 6.4.1 (Canary Exposure)**.** Given a canary $s[r]$, a model with parameters θ, and the randomness space $\mathcal{R}$, the exposure of $s[r]$ is

$$\text{exposure}_\theta = \log_2 |\mathcal{R}| - \log_2 \text{rank}_\theta(s[r])$$

After training, we calculate empirical model perplexity for all possibly-instantiated canaries and list them in sorted order. Then we can get the canary exposure based on the rank of a specific canary sequence $\text{rank}_\theta(s[r])$ and the number of all possible candidates $|\mathcal{R}|$. In our setting, we show the highest canary exposure in 10 canaries. For example, if a canary ranks 1st among 1M candidates, the canary exposure is 19.93.

Membership inference attack. Membership Inference is a widely used privacy attack method. Given a non-privately trained model, an adversary can predict whether or not a particular example was used to train the model. We adopt the membership inference attack in [158]. The general idea is to calculate the given samples' perplexities under the model, rank them and choose the ones with the lowest perplexities, i.e., highest likelihood by the model. We can think of this process as training a binary classifier based on the perplexity feature. We also implement the group membership inference attack to show the group confidentiality. More details about the implementation can be found in the

Appendix 6.5.5.

6.4.2 Overall performance

Figure 6.2 presents the results of model utilities and confidentiality guarantees across our models of interest on MultiWOZ 2.2 and CustomerSim datasets. Each point denotes a model for different epochs in a training process. Since the X-axis is ϵ in Bayesian Confidentiality (the lower the better) and the Y-axis is perplexity (the lower the better), a perfect model will lie in the bottom-left corner. CRT-GPT and DPSGD-GPT in general, perform better than S-DP-LSTM, CRT-LSTM and, DPSGD-LSTM on the test sets. Our model CRT-GPT's performance is close to Non-private-GPT in terms of PPL while preserving strong confidentiality. Besides, CRT-GPT is better than DPSGD-GPT manifested by a much lower ϵ, which demonstrates that approximately correct screening policy amplifies the confidentiality guarantee.

Differences can be witnessed in the results from two different datasets: the models trained on CustomerSim achieve overall better performances than those trained on MultiWOZ. We think it's due to the fact that CustomerSim contains simple dialogs from template-based simulations.

6.4.3 Attack results

Figure 6.3, 6.4, and 6.5 present the results from canary insertion attack and individual/group membership inference attack on MultiWOZ 2.2 and CustomerSim datasets. The X-axis is the false negative rate γ of screening policy π, ranging from 0.0 to 0.5; the Y-axis is the canary exposure (in Figure 6.3) and membership inference accuracy (in Figure 6.4 and 6.5), which measures the effectiveness of the attacks. The lower the canary exposure or inference accuracy, the better protection the model provides against

the attacks.

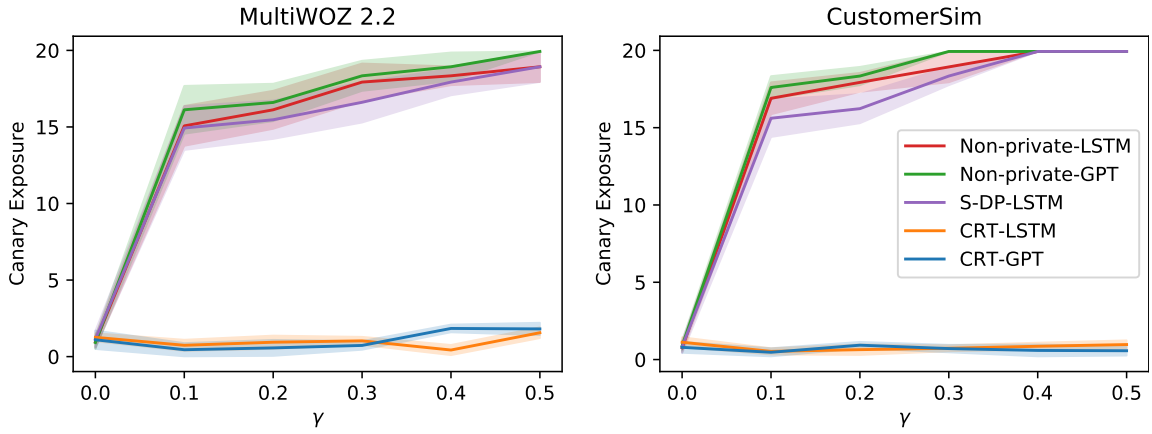

Figure 6.3: Canary insertion attack result. CRT achieves almost 0 canary exposure, which means it can prevent unintended memorization.

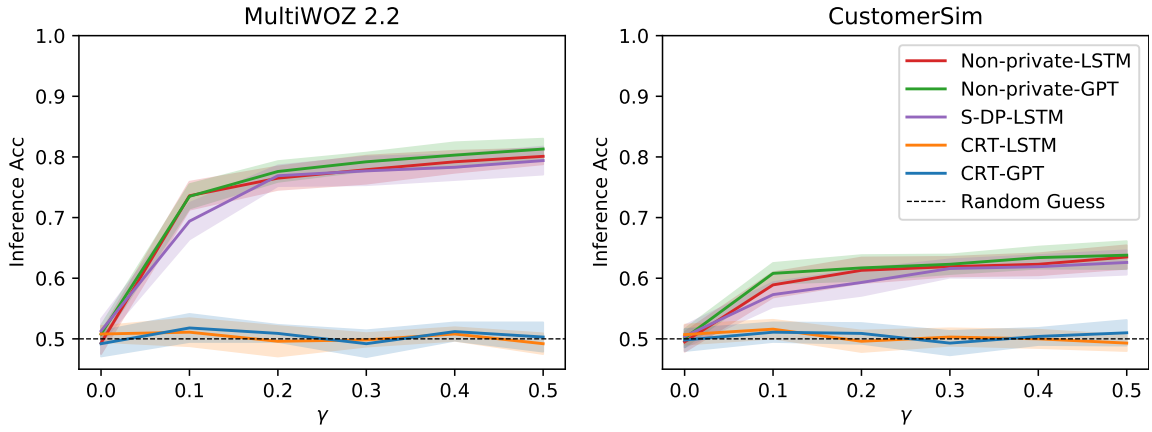

Figure 6.4: Membership inference attack result. CRT attains nearly 50% accuracy, indicating that the adversary could not infer whether a data point is used for training.

For canary insertion attack, it can be seen from Figure 6.3 that the canary exposures for CRT-LSTM and CRT-GPT are both close to 0 which thus guarantee excellent confidentiality. Non-private-LSTM and Non-private-GPT with mask can also attain great protection at perfect screening policy accuracy ($\gamma = 0$), nonetheless a rise in γ results in a sharp increase in the exposure. It should be noticed that S-DP-LSTM also has high exposure, similar to Non-private models, given any γ above 0. This is because that

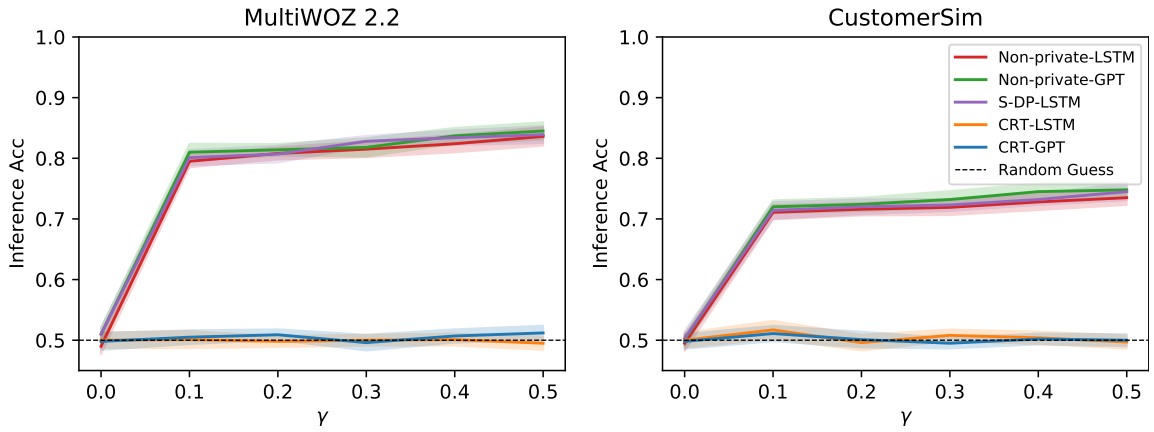

Figure 6.5: Group membership inference attack result.

many sensitive data has been falsely identified as non-sensitive by the approximate policy, S-DPSGD does not protect these false negative samples and hence a privacy leakage.

For membership inference attack, we compare the inference accuracy with the benchmark value of 0.5, which equals the random guess performance. In Figure 6.4 and 6.5, we see that CRT-LSTM and CRT-GPT align well with the 0.5 horizontal line, suggesting that they are rather safe to the attack. The inference accuracy for Non-private-LSTM/Non-private-GPT/S-DP-LSTM, in contrast, surges above 0.5 as the false negative rate γ deviates from 0.0, indicating that these models become vulnerable to the attack under non-perfect screen policy. In addition, Non-private and S-DP models show even worse protection under the group attack than the individual one in view of a higher inference accuracy at certain γ.

6.4.4 CRT amplifies Bayesian Confidentiality guarantees

Figure 6.6 shows that confidentially redacted training can help to amplify the confidentiality guarantees. We set the ϵ' in DP-SGD fixed and show the corresponding ϵ in Bayesian Confidentiality with different screen policy π. Both ϵ' and ϵ are for $\delta = 8e-5$. If the approximately screening policy π has a high recall (γ is small), we will achieve

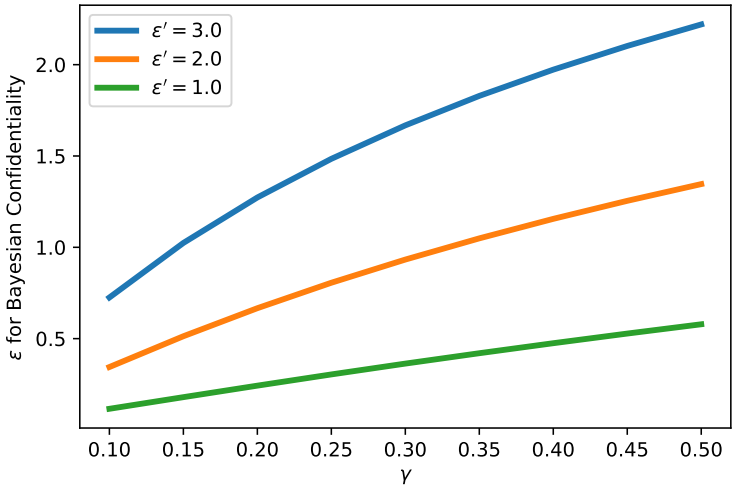

Figure 6.6: Bayesian Confidentiality amplification result. CRT helps to amplify the confidentiality guarantee.

much improvement in the Bayesian Confidentiality parameter ϵ by deduplication and redaction. For example, with $(\epsilon' = 1.0, \gamma = 0.1)$, we reduce the ϵ to 0.12.

6.5 Conclusion

In this chapter, we propose confidentially redacted training (CRT), a method to train language models while protecting secret texts. We introduce a new definition of confidentiality which quantifies the risk of leaking sensitive content. We prove the effectiveness of CRT both theoretically and empirically on multiple datasets and language models.

Broader Impact

This work will alleviate ethical concerns of large-scale pre-trained language models. This chapter provides one promising solution to an important aspect of NLP: training high quality language models for text generation without compromising confidential in-

formation. The current use cases of language models involve pretraining on public web corpus and fine-tuning on individual application data. However, the private application specific data often contains user-generated sensitive information. The proposed method in this chapter aims to use as much individual fine-tuning data as possible, while does not leak or memorize any confidential information with provable guarantees. Without the method, one has to either use the general pretraining LM without fine-tuning or manually filter sensitive information and fine-tuning on the remaining. It can be applied in broader applications that need language models or text generation models.

In our experiments, we use a simulation scheme to mimic confidential content in a real corpus. We did not compromise any real user's confidential information.

Additional Results and Proofs

6.5.1 Illustration of our proposed algorithm

6.5.2 Proofs of technical results

Proof of Proposition 6.3.5. The first statement straigtforwardly follows from that $\text{Redact}_\pi(D) = \text{Redact}_\pi(D')$ if $\pi(s, x) = 1$ and that $\text{Redact}_\pi(D)$ and $\text{Redact}_\pi(D')$ remain a pair of neighbors differing by only x. The group confidentiality claims follows from the standard calculation of small group privacy from differential privacy, which applies the (single x) confidentiality iteratively. Let $\tilde{D} = \text{Redact}_\pi(D)$, $\tilde{D}' = \text{Redact}_\pi(D')$ and $\tilde{S} = [x_1, ..., x_{\tilde{k}}]$

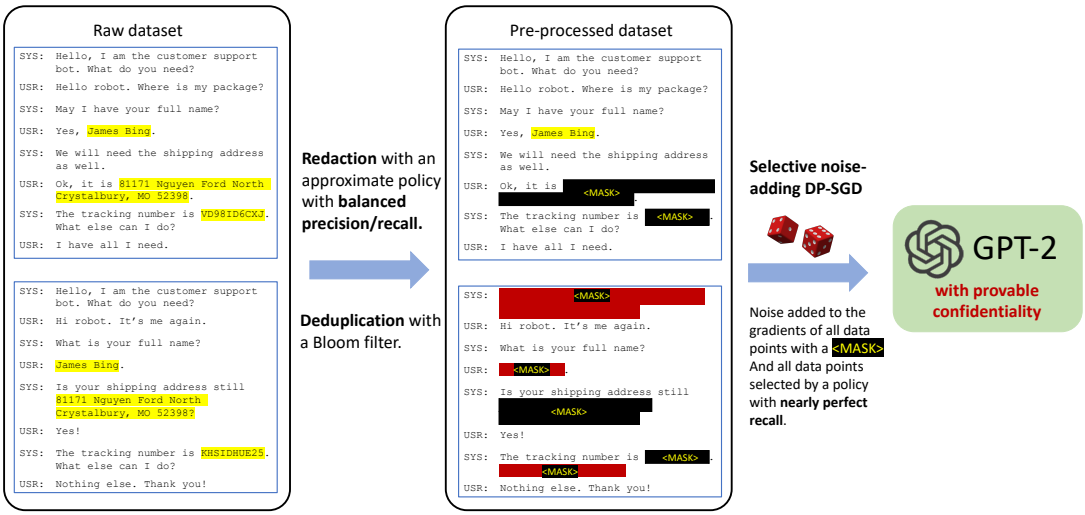

Figure 6.7: An illustration of our proposed algorithm on a dataset with two data points. The first data point is the example from Figure 6.1, and the second data point is modified to illustrate the various aspects of the pre-processing steps. The red-colored mask indicates the masks produced by deduplication just for illustration purposes. In the algorithm, both of them replace a sequence of tokens with the same special token `<MASK>`.

be the list of S that are not masked by π. For any measurable event E

$$\begin{aligned}
\mathbb{P}[\mathcal{A} \circ \text{Redact}_\pi(D) \in E] = \mathbb{P}[\mathcal{A}(\tilde{D})] &\leq e^{\epsilon x_1}\, \mathbb{P}[\mathcal{A}(\tilde{D}_{-x_1,+\texttt{<MASK>}}) \in E] + \delta \\
&\leq e^{\epsilon x_1 + \epsilon(x_2)}\, \mathbb{P}[\mathcal{A}(\tilde{D}_{-x_{1,2},+\texttt{<MASK>}^2}) \in E] + e^{\epsilon x_1}\delta + \delta \\
&\ldots \\
&\leq e^{\sum_{i=1}^{\tilde{k}} \epsilon x_i}\, \mathbb{P}[\mathcal{A}(\tilde{D}') \in E] + \delta(1 + e^{\epsilon x_1} + e^{\epsilon x_1 + \epsilon x_2} + \ldots + e^{\epsilon x_1 + \ldots + \epsilon x_{\tilde{k}-1}}) \\
&\leq e^{\tilde{\epsilon}(S)}\, \mathbb{P}[\mathcal{A} \circ \text{Redact}_\pi(D') \in E] + k e^{\tilde{\epsilon}(S)}\delta
\end{aligned}$$

□

Proof of Proposition 6.3.6. Consider a dataset D (in which one of the data point has

$x \sim \mu$) and a fixed D'. Denote the probability distributions p, q, r as shorthands for

$$p \sim \mathcal{A} \circ \text{Redact}_\pi(D) | \pi(s, x) = 1$$
$$q \sim \mathcal{A} \circ \text{Redact}_\pi(D) | \pi(s, x) = 0$$
$$r \sim \mathcal{A} \circ \text{Redact}_\pi(D') | \pi(s, x) = 0$$

Moreover, we use $\alpha p + (1 - \alpha)q$ to denote the mixture distribution that samples from p with probability α and q with probability $1 - \alpha$.

Recall that the Hockey-Stick-divergence characterization of (ϵ, δ)-indistinguishsability [178], which says that (P, Q) are (ϵ, δ)-indistinguishsable if and only if

$$H_{e^\epsilon}(P \| Q) := \mathop{\mathbb{E}}_{y \sim Q}[(\frac{dP}{dQ}(y) - e^\epsilon)_+] \leq \delta.$$

It suffices for us to bound the following quantity:

$$H_{1+\gamma(e^\epsilon - 1)}(\mathcal{A} \circ \text{Redact}_\pi(D) \| \mathcal{A} \circ \text{Redact}_\pi(D')) = H_{e^\epsilon}((1-\gamma)p + \gamma q \| (1-\gamma)p + \gamma r)$$
$$= \gamma H_{e^\epsilon}(q \| (1-\beta)p + \beta r) \leq \gamma \left((1-\beta) H_{e^\epsilon}(q \| p) + \beta H_{e^\epsilon}(q \| r)\right)$$

where $\beta = \frac{1+\gamma(e^\epsilon - 1)}{e^\epsilon}$. In the above, the second line follows from Theorem 2 of [179] (an identity called "Advanced Joint Convexity" by the authors) and the inequality is due to the (standard) joint convexity of the Hockey-Stick divergence. It remains to bound $H_{e^\epsilon}(q \| p)$ and $H_{e^\epsilon}(q \| r)$.

Check that $p, r, \mathcal{A} \circ \text{Redact}_\pi(D')$ are identically distributed and that $H_{e^\epsilon}(q \| r) \leq \delta$ by our assumption on $\mathcal{A}$'s Bayesian confidentiality guarantee w.r.t. $\mu(x | \pi(s, x) = 0)$. This completes the proof. □

Proof of Proposition 6.3.7. The proof is straightforward as $\text{Dedup}(D)$ differs from $\text{Dedup}(D')$

only by Unique(S). □

Proof of Theorem 6.3.8. The proof for the first statement follows from the fact that DP implies (ϵ, δ)-confidentiality and Proposition 6.3.5. Notably, if π_c catches all x that is missed by π, then we get that for all secret x, $\epsilon(x) \leq \epsilon$.

The proof of the second statement applies Proposition 6.3.7 and the second part of Proposition 6.3.5.

The proof of the third statement applies Proposition 6.3.6 but requires a separate treatment of the case when x is missed by both π and π_c. Let the event that a secret x is not selected by the conservative policy be E and let $\mathcal{A}$ be a generic algorithm satisfying (ϵ, δ_1) Bayesian confidentiality under μ,

$$\begin{aligned}\mathbb{P}[\mathcal{A}(D) \in S] &\leq \mathbb{P}[\mathcal{A} \circ \text{Redact}_\pi(D) \in S \subset E^c] + \delta \\ &\leq e^\epsilon \, \mathbb{P}[\mathcal{A}(D') \in S \subset E^c] + \delta_1 + \delta_2 \\ &\leq e^\epsilon \, \mathbb{P}[\mathcal{A}(D') \in S] + \delta_1 + \delta_2.\end{aligned}$$

This completes the proof. □

6.5.3 More details on experiments

We choose the one-layer LSTM with an embedding size of 200 and a hidden size of 200. We choose distill-gpt2[6] as the GPT-2 model, which has 6 layers, 768 dimension and 12 heads. Vocabulary size for GPT-2 is 50257. Our experiments are conducted on NVIDIA TITAN-Xp GPU. For LSTM models, we tune the hyperparameters of the learning rate (lr) among {20, 10, 5, 1, 0.1, 0.05, 0.01}, batch size (bs) and the epochs among {5, 10, 30, 50, 100}. We finally choose {lr=20, bs=256, epochs=50} for Non-private-LSTM,

[6]https://huggingface.co/distilgpt2

{lr=0.1, bs=5, epochs=50} for S-DPSGD-LSTM and {lr=0.05, bs=10, epochs=100} for CRT-LSTM. The same set of hyperparameters are tuned for GPT model as well. Our final choice for DPSGD-GPT/CRT-GPT model is {lr=5e-4, bs=256, epochs=10}. The actual run-time of algorithms depends on implementation details. Here, we outline estimates of the run-time for training. Running one epoch on CRT-LSTM takes 2 hours wheras the same task on CRT-GPT only takes 30 minutes since the implementation of [177] is highly efficient. We use autodp[7], an automating differential privacy computation for the privacy analysis. Noise scale σ is calculated numerically so that a DP budget of (ϵ, δ) is spent after T epochs.

6.5.4 Redaction policy details

We build the sequence labeling policy based on trimming one NER model[8] trained on OntoNotes-5.0 [180] dataset. We modify the last layer of the NER model and set the threshold for the output scores to enable abnormal/sensitive data detection. For the screen policy π, we set the threshold to be 0.3 for all predictions with OntoNotes tags. For the conservative policy π_c, we select all predictions with tags and all plain texts with scores smaller than 0.9 to be sensitive data. We manually label 200 data points and find that the conservative policy π_c can achieve 100% recall with lots of false positives and that π can achieve 90% recall with few false positives.

6.5.5 Membership inference attack details

In our experiments, we manually construct a dataset with 2000 sequences. We select 1000 sequences from the protected secrets used in the training data. And we randomly generate 1000 samples of similar format which are not used in the training data. In this

[7]https://github.com/yuxiangw/autodp
[8]https://huggingface.co/flair/ner-english-ontonotes-fast

way, a random guess generates an accuracy of 50%. For MultiWoz 2.2, we use sentences with reference numbers as the secrets. For CustomerSim, we choose customer addresses as the secrets.

In order to show group confidentiality guarantees, we also conduct group membership inference attack. In this setting, we construct a dataset with 2000 groups, each of which includes 20 sentences. One half of the groups are "sensitive groups" with all 20 sentences drawn from protected secrets and the other half are "insensitive groups" with all 20 sentences being random. We build the classifier based on the sum of the perplexities in one group.

6.5.6 "The devil is in the details" – how things could go wrong with seemingly inocuous changes to the algorithm.

In this section, we highlight various aspects of our algorithms and why certain choices in the pre-processing steps need to be done in the specific way we recommend for our results to hold for them.

1. It is important that the definition of confidentiality is defined with respect to a perfectly redacted version of the dataset. If we define it as in selective differential privacy, then there will *not* be an amplification effect from redaction. This is because if we replace a secret x that can be detected by π with another x' that cannot be detected by π, then even if x is replaced with `<MASK>`, x' will not be and the two datasets are still different after redaction. In addition, the S-DP definition will not be useful for us we do not know how to define a confidentiality parameter specific for each x or Bayesian confidentiality parameter for each μ

2. Tokenization and splitting into individual "sentences" (data points) should go before redaction / de-duplication. Otherwise redaction with an approximate screen-

ing policy and with an ideal screening policy, or deduplication may cause *misalignments*, resulting in almost all data points being different in the preprocessed version of D and D'.

3. Each data point should contain only "whole" natural sentences, otherwise the sensitive part of a natural sentence could split into two data points.

4. Deduplication steps should replace duplicate text with the same `<MASK>`, otherwise `<MASK_Dedup>` and `<MASK_Redact>` are not the same so even if all secrets are masked, there will be a difference between the pre-processed versions of D and its neighbor, while in our approach there are no differences and we achieve perfect confidentility (with $\epsilon = 0$).

5. Any data point containing `<MASK>` needs to be put in D^{pri}. This is because otherwise our algorithm that works on D' will be a deterministic algorithm that is perfectly distinguishable from the alternative world where the algorithm is random because the approximate policy π fails to redact certain secrets x.

6. In the DP-SGD algorithm, the sampled minibatches should contain the whole minibatch from D^{pri} or the whole minibatch from D^{pub}. Otherwise the noise always need to be added and the algorithm is identical to the vanilla DP-SGD, and there is no benefit of having a portion of the data being public comparing to all of the data are private.

Chapter 7

Conclusion and Future Work

This thesis has explored and addressed some of the most pressing challenges in empowering the responsible use of Large Language Models, focusing on the realms of safety, security, and privacy. As the rapid advancement of generative AI technologies like ChatGPT and Llama continues to shape the technological landscape, the risks and ethical concerns surrounding their misuse become increasingly significant. Our work has sought to mitigate these risks through a series of novel techniques, providing theoretical guarantees and empirical evidence across three critical areas: watermarking, intellectual property protection, and privacy preservation.

In the domain of watermarking LLM-generated text, we have proposed two novel solutions. The Unigram-Watermark method provides a rigorous theoretical framework to quantify performance drops, detection accuracy, and security against post-processing while ensuring that the watermarked LLM remains close to the original. The PF Watermark, building upon the Permute-and-Flip decoding method, demonstrates Pareto optimality in balancing robustness and perplexity while effectively detecting watermarked text.

To protect the intellectual property of generative AI, we have developed innovative

watermarking techniques tailored for LLMs. The Distillation-Resistant Watermarking (DRW) method protects NLP models from being stolen via distillation by injecting watermarks into the victim's prediction probabilities corresponding to a secret key. Ginsew, an adaptation of DRW principles to text generation models, injects secret watermarking signals into decoding steps to ascertain the origin of the model, providing a formidable defense against unauthorized distillation.

Addressing the privacy risks posed by LLMs, we have proposed the Confidentially Redacted Training (CRT) method. Inspired by differential privacy, CRT trains language generation models while protecting confidential segments by randomizing parts of the training process. We have demonstrated that redaction with an approximately correct screening policy amplifies the confidentiality guarantee, helping mitigate privacy risks in the initial model training phase.

In conclusion, this thesis has made significant contributions to the responsible and ethical development of LLMs by introducing:

1. Watermarking techniques for detecting AI-generated text with theoretical guarantees and empirical validation.

2. Watermarking methods to protect the intellectual property of LLMs against model-stealing attacks.

3. Privacy-preserving training techniques that prevent LLMs from generating sensitive information.

These contributions form a comprehensive framework for ensuring the safe, secure, and private deployment of LLMs, ultimately helping to align these powerful technologies with societal values.

I aim to focus on advancing trustworthy, responsible, and human-centric AI in the future. Some key directions I plan to pursue include:

Multimodal watermarking and misuse monitoring. Current watermarking methods are domain-specific, often limited to text or images. However, as demonstrated by models like GPT-4V, future AI systems will likely be multimodal, integrating capabilities across image, text, and audio modalities. This multimodality is crucial for achieving Artificial General Intelligence (AGI). However, it also poses a central challenge for AI safety: detecting AI-generated content and verifying official content. Building on my expertise in watermarking, I aim to develop next-generation watermarks tailored to multimodal foundation models. For instance, I plan to explore using the same pseudorandom function to control the generation process across modalities, such that different modalities mutually reinforce detection confidence. Furthermore, I intend to research monitoring and mitigating the misuse of AI-generated content. Monitoring the misuse of AI-generated content has significant societal benefits. For example, it can help detect cheating in educational settings and guide students accordingly. It can also uncover AI-enabled fraud and deception. My goal is to pioneer watermarking and monitoring techniques specific to emerging multimodal AI that promote responsible and ethical use of them.

Exhaustive evaluations and red teaming for safety alignment. As more powerful AI systems emerge, we need incrementally scaled frameworks to rigorously assess and mitigate new risks. In the near future, the most capable AI may have access to databases, operating systems, and even robotics, bringing more challenges for model safety. For instance, we must prevent language models from executing harmful commands like "`rm -rf /*`" to delete data. Interactions with the physical world through robotics pose further dangers. Thoroughly testing models for risks is crucial before deployment. I plan to do research on scalable evaluation protocols and techniques for automatic red teaming of these emerging AI models. On the attacking side, I intend to explore how to automatically assess risks to national security and public health and safety. This

will involve investigating adversarial example generation and organizing model hacking competitions to identify vulnerabilities. On the defense side, the focus will be on ensuring that models align with human values, benefit the broader society, and are robust against adversarial attacks, such as those aiming to "jailbreak" the models.

Privacy enhancing technologies, regulation, and policy. The rapid advancement of AI models underscores the increasing significance of training data memorization and associated privacy concerns. Recognizing this, the US government has enacted regulations to scrutinize how agencies collect and use commercially available information. In response to these mounting privacy concerns, I intend to leverage my previous experience in privacy-preserving generative AI. My objective is to incorporate differentially private and federated learning techniques at every stage of AI model development, which includes pre-training, fine-tuning, and inference phases. Additionally, I plan to conduct research on auditing privacy leakage. My internship experience with the Google Privacy Research team has also equipped me to utilize AI for privacy, such as using modern NLP models to identify privacy issues from user feedback on products. Moving forward, as regulations surrounding AI continue to evolve, I am eager to collaborate with researchers on AI regulation and policy topics, such as AI safety standards, model risk ratings, and transparency requirements. This research could provide policymakers and regulators with technical expertise.

In summary, I aim to advance trustworthy AI that respects human values through open and reproducible research. My vision is to increase public understanding of responsible innovation and engage wider audiences in shaping the future trajectory of AI.

Bibliography

[1] OpenAI, *Chatgpt: Optimizing language models for dialogue*, *OpenAI blog* (2022).

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et. al.*, *Llama 2: Open foundation and fine-tuned chat models*, *arXiv preprint arXiv:2307.09288* (2023).

[3] The White House, "Executive order on the safe, secure, and trustworthy development and use of artificial intelligence." https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artif 2023.

[4] AI Safety Summit, "Ai safety summit 2023." https://www.gov.uk/government/topical-events/ai-safety-summit-2023, 2023.

[5] AISIC, "U.s. artificial intelligence safety institute." https://www.nist.gov/aisi, 2023.

[6] X. Zhao, P. Ananth, L. Li, and Y.-X. Wang, *Provable robust watermarking for ai-generated text*, *ICLR 2024* (2024).

[7] X. Zhao, L. Li, and Y.-X. Wang, *Permute-and-flip: An optimally robust and watermarkable decoder for llms*, *arXiv preprint arXiv:2402.05864* (2024).

[8] X. Zhao, L. Li, and Y.-X. Wang, *Distillation-resistant watermarking for model protection in nlp*, in *Conference on Empirical Methods in Natural Language Processing*, 2022.

[9] X. Zhao, Y.-X. Wang, and L. Li, *Protecting language generation models via invisible watermarking*, *arXiv preprint arXiv:2302.03162* (2023).

[10] X. Zhao, L. Li, and Y.-X. Wang, *Provably confidential language modelling*, in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 943–955, 2022.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et. al.*, *Language models are few-shot learners*, *Advances in neural information processing systems* **33** (2020) 1877–1901.

[12] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical text-conditional image generation with clip latents*, *ArXiv* **abs/2204.06125** (2022).

[13] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, *et. al.*, *Photorealistic text-to-image diffusion models with deep language understanding*, *Advances in Neural Information Processing Systems* **35** (2022) 36479–36494.

[14] OpenAI, *Gpt-4 technical report*, *ArXiv* **abs/2303.08774** (2023).

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

[16] L. Weidinger, J. F. J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, Z. Kenton, S. M. Brown, W. T. Hawkins, T. Stepleton, C. Biles, A. Birhane, J. Haas, L. Rimell, L. A. Hendricks, W. S. Isaac, S. Legassick, G. Irving, and I. Gabriel, *Ethical and social risks of harm from language models*, *ArXiv* **abs/2112.04359** (2021).

[17] C. Stokel-Walker, *Ai bot chatgpt writes smart essays - should professors worry?*, *Nature* (2022).

[18] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, *Robust speech recognition via large-scale weak supervision*, *ArXiv* **abs/2212.04356** (2022).

[19] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, *Poisoning web-scale training datasets is practical*, *ArXiv* **abs/2302.10149** (2023).

[20] A. M. Turing, *Computing machinery and intelligence*. 1950.

[21] S. Gehrmann, H. Strobelt, and A. M. Rush, *Gltr: Statistical detection and visualization of generated text*, in *Annual Meeting of the Association for Computational Linguistics*, 2019.

[22] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, *Detectgpt: Zero-shot machine-generated text detection using probability curvature*, *ArXiv* **abs/2301.11305** (2023).

[23] D. Hovy, *The enemy in your own camp: How well can we detect statistically-generated fake reviews – an adversarial study*, in *Annual Meeting of the Association for Computational Linguistics*, 2016.

[24] OpenAI, *New ai classifier for indicating ai-written text*, *OpenAI blog* (2023).

[25] W. Liang, M. Yuksekgonul, Y. Mao, E. Wu, and J. Y. Zou, *Gpt detectors are biased against non-native english writers*, *ArXiv* **abs/2304.02819** (2023).

[26] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, *A watermark for large language models*, *International Conference on Machine Learning* (2023).

[27] S. Aaronson, *Simons institute talk on watermarking of large language models*, 2023.

[28] M. Christ, S. Gunn, and O. Zamir, *Undetectable watermarks for language models*, *arXiv preprint arXiv:2306.09194* (2023).

[29] M. Albert, *Concentration inequalities for randomly permuted sums*, in *High Dimensional Probability VIII: The Oaxaca Volume*, pp. 341–383, Springer, 2019.

[30] K. Stefan, A. Fabien, *et. al.*, *Information hiding techniques for steganography and digital watermarking*, 2000.

[31] U. Topkara, M. Topkara, and M. J. Atallah, *The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions*, in *Workshop on Multimedia & Security*, 2006.

[32] M. J. Atallah, V. Raskin, M. Crogan, C. F. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik, *Natural language watermarking: Design, analysis, and a proof-of-concept implementation*, in *Information Hiding*, 2001.

[33] M. J. Atallah, V. Raskin, C. F. Hempelmann, M. Topkara, R. Sion, U. Topkara, and K. E. Triezenberg, *Natural language watermarking and tamperproofing*, in *Information Hiding*, 2002.

[34] X. Yang, J. Zhang, K. Chen, W. Zhang, Z. Ma, F. Wang, and N. Yu, *Tracing text provenance via context-aware lexical substitution*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 11613–11621, 2022.

[35] H. Ueoka, Y. Murawaki, and S. Kurohashi, *Frustratingly easy edit-based linguistic steganography with a masked language model*, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[36] M. Gambini, T. Fagni, F. Falchi, and M. Tesconi, *On pushing deepfake tweet detection capabilities to the limits*, *Proceedings of the 14th ACM Web Science Conference 2022* (2022).

[37] M. Wolff, *Attacking neural text detectors*, *ArXiv* **abs/2002.11768** (2020).

[38] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi, *Can ai-generated text be reliably detected?*, *ArXiv* **abs/2303.11156** (2023).

[39] S. Chakraborty, A. S. Bedi, S. Zhu, B. An, D. Manocha, and F. Huang, *On the possibilities of ai-generated text detection*, *arXiv preprint arXiv:2304.04736* (2023).

[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, *OpenAI blog* (2019).

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, *Advances in neural information processing systems* **30** (2017).

[42] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, *ArXiv* **abs/1907.11692** (2019).

[43] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, in *Theory of cryptography conference*, pp. 265–284, Springer, 2006.

[44] J. Dong, D. Durfee, and R. Rogers, *Optimal differential privacy composition for exponential mechanisms*, in *International Conference on Machine Learning*, pp. 2597–2606, PMLR, 2020.

[45] K. Krishna, Y. Song, M. Karpinska, J. Wieting, and M. Iyyer, *Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense*, *ArXiv* **abs/2303.13408** (2023).

[46] S. Merity, C. Xiong, J. Bradbury, and R. Socher, *Pointer sentinel mixture models*, in *International Conference on Learning Representations*, 2017.

[47] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, *Opt: Open pre-trained transformer language models*, *ArXiv* **abs/2205.01068** (2022).

[48] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, *Llama: Open and efficient foundation language models*, *ArXiv* **abs/2302.13971** (2023).

[49] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, *The curious case of neural text degeneration*, in *International Conference on Learning Representations*, 2020.

[50] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, *Huggingface's transformers: State-of-the-art natural language processing*, *ArXiv* **abs/1910.03771** (2019).

[51] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe, *Training language models to follow instructions with human feedback*, *ArXiv* **abs/2203.02155** (2022).

[52] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. rahman Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, in *Annual Meeting of the Association for Computational Linguistics*, 2019.

[53] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, *Teaching machines to read and comprehend*, *Advances in neural information processing systems* **28** (2015).

[54] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, *Exploring the limits of transfer learning with a unified text-to-text transformer*, *The Journal of Machine Learning Research* **21** (2020), no. 1 5485–5551.

[55] F. McSherry and K. Talwar, *Mechanism design via differential privacy*, in *Symposium on Foundations of Computer Science (FOCS'07)*, pp. 94–103, IEEE, 2007.

[56] M. Cesar and R. Rogers, *Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics*, in *Algorithmic Learning Theory*, pp. 421–457, PMLR, 2021.

[57] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, *et. al.*, *Constitutional ai: Harmlessness from ai feedback*, *arXiv preprint arXiv:2212.08073* (2022).

[58] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, *The curious case of neural text degeneration*, in *International Conference on Learning Representations*, 2019.

[59] D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch, *Comparison of diverse decoding methods from conditional language models*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3752–3762, 2019.

[60] G. Wiher, C. Meister, and R. Cotterell, *On decoding strategies for neural text generators*, *Transactions of the Association for Computational Linguistics* **10** (2022) 997–1012.

[61] Y. Chen, S. Gilroy, A. Maletti, J. May, and K. Knight, *Recurrent neural networks as weighted language recognizers*, *ArXiv* **abs/1711.05408** (2017).

[62] X. Lu, S. Welleck, P. West, L. Jiang, J. Kasai, D. Khashabi, R. L. Bras, L. Qin, Y. Yu, R. Zellers, *et. al.*, *Neurologic a* esque decoding: Constrained text generation with lookahead heuristics*, *arXiv preprint arXiv:2112.08726* (2021).

[63] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et. al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, *Advances in Neural Information Processing Systems* **33** (2020) 9459–9474.

[64] C. I. Meister, E. Salesky, and R. Cotterell, *Generalized entropy regularization or: There's nothing special about label smoothing*, in *Annual Meeting of the Association for Computational Linguistics (ACL-2020)*, pp. 6870–6886, Association for Computational Linguistics (ACL), 2020.

[65] R. Kuditipudi, J. Thickstun, T. Hashimoto, and P. Liang, *Robust distortion-free watermarks for language models*, *ArXiv* **abs/2307.15593** (2023).

[66] Z. Zhang, L. Lyu, W. Wang, L. Sun, and X. Sun, *How to inject backdoors with better consistency: Logit anchoring on clean data*, *arXiv preprint arXiv:2109.01300* (2021).

[67] J. Lin, L. Dang, M. Rahouti, and K. Xiong, *Ml attack models: adversarial attacks and data poisoning attacks*, *arXiv preprint arXiv:2112.02797* (2021).

[68] H. Zhang, Z. Guo, H. Zhu, B. Cao, L. Lin, J. Jia, J. Chen, and D. Wu, *On the safety of open-sourced large language models: Does alignment really prevent them from being misused?*, *ArXiv* **abs/2310.01581** (2023).

[69] X. Zhao, X. Yang, T. Pang, C. Du, L. Li, Y.-X. Wang, and W. Y. Wang, *Weak-to-strong jailbreaking on large language models*, *arXiv preprint arXiv:2401.17256* (2024).

[70] R. McKenna and D. R. Sheldon, *Permute-and-flip: A new mechanism for differentially private selection*, *Advances in Neural Information Processing Systems* **33** (2020) 193–203.

[71] Z. Ding, D. Kifer, T. Steinke, Y. Wang, Y. Xiao, D. Zhang, *et. al.*, *The permute-and-flip mechanism is identical to report-noisy-max with exponential noise*, *arXiv preprint arXiv:2105.07260* (2021).

[72] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, vol. 33. US Government Printing Office, 1948.

[73] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model." https://github.com/tatsu-lab/stanford_alpaca, 2023.

[74] P. Zhang, G. Zeng, T. Wang, and W. Lu, *Tinyllama: An open-source small language model*, 2024.

[75] K. Pillutla, S. Swayamdipta, R. Zellers, J. Thickstun, S. Welleck, Y. Choi, and Z. Harchaoui, *Mauve: Measuring the gap between neural text and human text using divergence frontiers*, *Advances in Neural Information Processing Systems* **34** (2021) 4816–4828.

[76] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, *Neural text generation with unlikelihood training*, *arXiv preprint arXiv:1908.04319* (2019).

[77] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, and D. Batra, *Diverse beam search: Decoding diverse solutions from neural sequence models*, *arXiv preprint arXiv:1610.02424* (2016).

[78] P. Anderson, B. Fernando, M. Johnson, and S. Gould, *Guided open vocabulary image captioning with constrained beam search*, *arXiv preprint arXiv:1612.00576* (2016).

[79] L. Qin, V. Shwartz, P. West, C. Bhagavatula, J. D. Hwang, R. Le Bras, A. Bosselut, and Y. Choi, *Backpropagation-based decoding for unsupervised counterfactual and abductive reasoning*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 794–805, 2020.

[80] C. Hokamp and Q. Liu, *Lexically constrained decoding for sequence generation using grid beam search*, *arXiv preprint arXiv:1704.07138* (2017).

[81] X. L. Li, A. Holtzman, D. Fried, P. Liang, J. Eisner, T. Hashimoto, L. Zettlemoyer, and M. Lewis, *Contrastive decoding: Open-ended text generation as optimization*, *arXiv preprint arXiv:2210.15097* (2022).

[82] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, *Accelerating large language model decoding with speculative sampling*, *arXiv preprint arXiv:2302.01318* (2023).

[83] E. Tulchinskii, K. Kuznetsov, L. Kushnareva, D. Cherniavskii, S. Barannikov, I. Piontkovskaya, S. I. Nikolenko, and E. Burnaev, *Intrinsic dimension estimation for robust detection of ai-generated texts*, *ArXiv* **abs/2306.04723** (2023).

[84] X. Yang, W. Cheng, L. Petzold, W. Y. Wang, and H. Chen, *Dna-gpt: Divergent n-gram analysis for training-free detection of gpt-generated text*, *ArXiv* **abs/2305.17359** (2023).

[85] G. Bao, Y. Zhao, Z. Teng, L. Yang, and Y. Zhang, *Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature*, *ArXiv* **abs/2310.05130** (2023).

[86] GPTZero, *Gptzero: More than an ai detector preserve what's human.*, *GPTZero website* (2023).

[87] ZeroGPT, *Zerogpt: Trusted gpt-4, chatgpt and ai detector tool by zerogpt*, *ZeroGPT website* (2023).

[88] Y. Chen, H. Kang, V. Zhai, L. Li, R. Singh, and B. Ramakrishnan, *Gpt-sentinel: Distinguishing human and chatgpt generated content*, *ArXiv* **abs/2305.07969** (2023).

[89] X. Yu, Y. Qi, K. Chen, G. Chen, X. Yang, P. Zhu, W. Zhang, and N. H. Yu, *Gpt paternity test: Gpt generated text detection with gpt genetic inheritance*, *ArXiv* **abs/2305.12519** (2023).

[90] X. Liu, Z. Zhang, Y. Wang, Y. Lan, and C. Shen, *Coco: Coherence-enhanced machine-generated text detection under data limitation with contrastive learning*, *ArXiv* **abs/2212.10341** (2022).

[91] X. Hu, P.-Y. Chen, and T.-Y. Ho, *Radar: Robust ai-text detection via adversarial learning*, *ArXiv* **abs/2307.03838** (2023).

[92] Z. Shi, Y. Wang, F. Yin, X. Chen, K.-W. Chang, and C.-J. Hsieh, *Red teaming language model detectors with language models*, *ArXiv* **abs/2305.19713** (2023).

[93] Z. Hu, L. Chen, X. Wu, Y. Wu, H. Zhang, and H. Huang, *Unbiased watermark for large language models*, *ArXiv* **abs/2310.10669** (2023).

[94] Y. Wu, Z. Hu, H. Zhang, and H. Huang, *Dipmark: A stealthy, efficient and resilient watermark for large language models*, *ArXiv* **abs/2310.07710** (2023).

[95] K. Yoo, W. Ahn, J. Jang, and N. J. Kwak, *Robust multi-bit natural language watermarking through invariant features*, in *Annual Meeting of the Association for Computational Linguistics*, 2023.

[96] P. Fernandez, A. Chaffin, K. Tit, V. Chappelier, and T. Furon, *Three bricks to consolidate watermarks for large language models*, *2023 IEEE International Workshop on Information Forensics and Security (WIFS)* (2023) 1–6.

[97] Y. Zhu and Y.-X. Wang, *Adaptive private-k-selection with adaptive k and application to multi-label pate*, in *International Conference on Artificial Intelligence and Statistics*, pp. 5622–5635, PMLR, 2022.

[98] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, in *NAACL*, 2019.

[99] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *RoBERTa: A robustly optimized BERT pretraining approach*, *arXiv preprint arXiv:1907.11692* (2019).

[100] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, *Stealing machine learning models via prediction apis*, in *USENIX Security*, 2016.

[101] E. Wallace, M. Stern, and D. X. Song, *Imitation attacks and defenses for black-box machine translation systems*, in *Conference on Empirical Methods in Natural Language Processing*, 2020.

[102] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, *Thieves on sesame street! model extraction of bert-based apis*, in *ICLR*, 2020.

[103] X. He, L. Lyu, Q. Xu, and L. Sun, *Model extraction and adversarial transferability, your bert is vulnerable!*, in *NAACL*, 2021.

[104] T. Orekondy, B. Schiele, and M. Fritz, *Knockoff nets: Stealing functionality of black-box models*, in *CVPR*, 2019.

[105] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, *Dawn: Dynamic adversarial watermarking of neural networks*, *Proceedings of the 29th ACM International Conference on Multimedia* (2021).

[106] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, *Entangled watermarks as a defense against model extraction*, in *USENIX Security*, 2021.

[107] L. Charette, L. Chu, Y. Chen, J. Pei, L. Wang, and Y. Zhang, *Cosine model watermarking against ensemble distillation*, *AAAI* (2022).

[108] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, *Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data*, in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[109] G. E. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, *ArXiv* **abs/1503.02531** (2015).

[110] M. Phuong and C. Lampert, *Towards understanding knowledge distillation*, in *ICML*, 2019.

[111] H. Zhou, L. Song, J. Chen, Y. Zhou, G. Wang, J. Yuan, and Q. Zhang, *Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective*, in *ICLR*, 2021.

[112] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, *Embedding watermarks into deep neural networks*, *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (2017).

[113] Y. Adi, C. Baum, M. Cissé, B. Pinkas, and J. Keshet, *Turning your weakness into a strength: Watermarking deep neural networks by backdooring*, in *USENIX Security*, 2018.

[114] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, *Protecting intellectual property of deep neural networks with watermarking*, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security* (2018).

[115] E. L. Merrer, P. Pérez, and G. Trédan, *Adversarial frontier stitching for remote neural network watermarking*, *Neural Computing and Applications* **32** (2019) 9233–9244.

[116] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, *Prada: Protecting against dnn model stealing attacks*, *2019 IEEE European Symposium on Security and Privacy (EuroS&P)* (2019) 512–527.

[117] X. He, Q. Xu, L. Lyu, F. Wu, and C. Wang, *Protecting intellectual property of language generation apis with lexical watermark*, in *AAAI Conference on Artificial Intelligence*, 2021.

[118] J. D. Scargle, *Studies in astronomical time series analysis. ii - statistical aspects of spectral analysis of unevenly spaced data*, *The Astrophysical Journal* **263** (1982) 835–853.

[119] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Glue: A multi-task benchmark and analysis platform for natural language understanding*, in *BlackboxNLP@EMNLP*, 2018.

[120] E. T. K. Sang and F. D. Meulder, *Introduction to the conll-2003 shared task: Language-independent named entity recognition*, in *CoNLL*, 2003.

[121] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, in *EMNLP*, 2013.

[122] W. B. Dolan and C. Brockett, *Automatically constructing a corpus of sentential paraphrases*, in *IJCNLP*, 2005.

[123] J. Chen, J. Wang, T. Peng, Y. Sun, P. Cheng, S. Ji, X. Ma, B. Li, and D. Song, *Copy, right? a testing framework for copyright protection of deep learning models*, in *IEEE Symposium on Security and Privacy (SP)*, 2022.

[124] B. Fuglede and F. Topsøe, *Jensen-shannon divergence and hilbert space embedding*, *ISIT* (2004).

[125] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, in *ICLR*, 2019.

[126] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, *CoRR* **abs/1412.6980** (2015).

[127] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, in *Advances in Neural Information Processing Systems*, 2020.

[128] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, ch. 3. Cambridge university press, 2014.

[129] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, *Turning your weakness into a strength: Watermarking deep neural networks by backdooring*, in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[130] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, *Protecting intellectual property of deep neural networks with watermarking*, in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018.

[131] X. He, Q. Xu, Y. Zeng, L. Lyu, F. Wu, J. Li, and R. Jia, *Cater: Intellectual property protection on text generation apis via conditional watermarks*, in *Advances in Neural Information Processing Systems*, 2022.

[132] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, *Stealing machine learning models via prediction {APIs}*, in *25th USENIX security symposium (USENIX Security 16)*, 2016.

[133] T. Orekondy, B. Schiele, and M. Fritz, *Knockoff nets: Stealing functionality of black-box models*, in *Conference on Computer Vision and Pattern Recognition*, 2019.

[134] X. He, L. Lyu, Q. Xu, and L. Sun, *Model extraction and adversarial transferability, your bert is vulnerable!*, in *Conference of the North American Chapter of the Association for Computational Linguistics*, 2020.

[135] Q. Xu, X. He, L. Lyu, L. Qu, and G. Haffari, *Student surpasses teacher: Imitation attack for black-box nlp apis*, in *International Conference on Computational Linguistics*, 2021.

[136] E. L. Merrer, P. Pérez, and G. Trédan, *Adversarial frontier stitching for remote neural network watermarking*, *Neural Computing and Applications* (2017).

[137] L. Charette, L. Chu, Y. Chen, J. Pei, L. Wang, and Y. Zhang, *Cosine model watermarking against ensemble distillation*, in *AAAI Conference on Artificial Intelligence*, 2022.

[138] H. Jia, C. A. Choquette-Choo, and N. Papernot, *Entangled watermarks as a defense against model extraction*, in *30th USENIX security symposium (USENIX Security 21)*, 2021.

[139] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, *Report on the 11th iwslt evaluation campaign*, in *IWSLT*, 2014.

[140] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, *Findings of the 2014 workshop on statistical machine translation*, in *WMT@ACL*, 2014.

[141] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, in *Annual Meeting of the Association for Computational Linguistics*, 2002.

[142] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *Bertscore: Evaluating text generation with bert*, in *International Conference on Learning Representations*, 2019.

[143] R. Sennrich, B. Haddow, and A. Birch, *Neural machine translation of rare words with subword units*, in *Annual Meeting of the Association for Computational Linguistics*, 2016.

[144] N. Mostafazadeh, N. Chambers, X. He, D. Parikh, D. Batra, L. Vanderwende, P. Kohli, and J. F. Allen, *A corpus and cloze evaluation for deeper understanding of commonsense stories*, in *North American Chapter of the Association for Computational Linguistics*, 2016.

[145] C.-Y. Lin, *Rouge: A package for automatic evaluation of summaries*, in *Annual Meeting of the Association for Computational Linguistics*, 2004.

[146] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, *fairseq: A fast, extensible toolkit for sequence modeling*, in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[147] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *International Conference on Learning Representations*, 2015.

[148] C. D. Fellbaum, *Wordnet : an electronic lexical database*, *Language* **76** (2000) 706.

[149] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, in *International Conference on Learning Representations*, 2013.

[150] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin, *Convolutional sequence to sequence learning*, in *International Conference on Machine Learning*, 2017.

[151] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, *Multilingual denoising pre-training for neural machine translation*, *Transactions of the Association for Computational Linguistics* **8** (2020) 726–742.

[152] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, *CoRR* **abs/1409.0473** (2015).

[153] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, *A simple language model for task-oriented dialogue*, *ArXiv* **abs/2005.00796** (2020).

[154] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. V. Le, and S. Petrov, *Natural questions: A benchmark for question answering research*, *Transactions of the Association for Computational Linguistics* **7** (2019) 453–466.

[155] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones, *Word embedding based generalized language model for information retrieval*, *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015).

[156] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, *ArXiv* **abs/1706.03762** (2017).

[157] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. X. Song, *The secret sharer: Evaluating and testing unintended memorization in neural networks*, in *USENIX Security Symposium*, 2019.

[158] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. X. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, *Extracting training data from large language models*, in *USENIX Security Symposium*, 2021.

[159] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini, *Deduplicating training data makes language models better*, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2022.

[160] W. Shi, A. Cui, E. Li, R. Jia, and Z. Yu, *Selective differential privacy for language modeling*, *ArXiv* **abs/2108.12944** (2021).

[161] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, *ArXiv* **abs/1810.04805** (2019).

[162] J. Howard and S. Ruder, *Universal language model fine-tuning for text classification*, in *ACL*, 2018.

[163] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, *Exploring the limits of transfer learning with a unified text-to-text transformer*, *ArXiv* **abs/1910.10683** (2020).

[164] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural Computation* **9** (1997) 1735–1780.

[165] T. Mikolov, M. Karafiát, L. Burget, J. H. Cernocký, and S. Khudanpur, *Recurrent neural network based language model*, in *INTERSPEECH*, 2010.

[166] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, *Deep learning with differential privacy*, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[167] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, *Learning differentially private recurrent language models*, in *ICLR*, 2018.

[168] R. Anil, B. Ghazi, V. Gupta, R. Kumar, and P. Manurangsi, *Large-scale differentially private bert*, *ArXiv* **abs/2108.01624** (2021).

[169] V. Feldman, *Does learning require memorization? a short tale about a long tail*, in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 954–959, 2020.

[170] N. Kandpal, E. Wallace, and C. Raffel, *Deduplicating training data mitigates privacy risks in language models*, *arXiv preprint arXiv:2202.06539* (2022).

[171] A. Ghosh and A. Roth, *Selling privacy at auction*, *Games and Economic Behavior* **91** (2015) 334–346.

[172] A. Triastcyn and B. Faltings, *Bayesian differential privacy for machine learning*, in *International Conference on Machine Learning*, pp. 9583–9592, PMLR, 2020.

[173] B. H. Bloom, *Space/time trade-offs in hash coding with allowable errors*, *Communications of the ACM* **13** (1970), no. 7 422–426.

[174] X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, and J. Chen, *Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines*, in *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, pp. 109–117, 2020.

[175] T. Zhao and M. Eskénazi, *Zero-shot dialog generation with cross-domain latent actions*, in *SIGDIAL Conference*, 2018.

[176] K. Jarvinen, *Design and implementation of a sha-1 hash module on fpgas*, *Helsinki University of Technology Signal Processing Laboratory* (2004).

[177] X. Li, F. Tramèr, P. Liang, and T. Hashimoto, *Large language models can be strong differentially private learners*, *ArXiv* **abs/2110.05679** (2021).

[178] G. Barthe and F. Olmedo, *Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs*, in *International Colloquium on Automata, Languages, and Programming*, pp. 49–60, Springer, 2013.

[179] B. Balle, G. Barthe, and M. Gaboardi, *Privacy amplification by subsampling: tight analyses via couplings and divergences*, in *Advances in Neural Information Processing Systems*, pp. 6280–6290, 2018.

[180] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, and et al., *Ontonotes release 5.0*, *Linguistic Data Consortium, Philadelphia, PA, 23* (2013).