12-19-2023

Mitigating The Shortcomings of Language Models: Strategies For Handling Memorization & Adversarial Attacks

Aly Kassem
University of Windsor

Mitigating The Shortcomings of Language Models: Strategies For Handling Memorization & Adversarial Attacks

By

Aly Kassem

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2023

Mitigating The Shortcomings of Language Models: Strategies For Handling Memorization & Adversarial Attacks

by

Aly Kassem

APPROVED BY:

M. Mirhassani
Department of Electrical and Computer Engineering

L. Rueda
School of Computer Science

S. Saad, Advisor
School of Computer Science

November 1, 2023

DECLARATION OF CO-AUTHORSHIP AND PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is the result of joint research, as follows: Chapter 2 of the thesis was co-authored with Dr. Saad and Ph.D. student Omar Mahmoud. Chapter 3 was co-authored with Dr. Saad. In the case of Chapters 2 and 3, I conceived the research idea, set the theoretical framework, designed and executed the experiments, supervised data collection and analysis, and led the manuscript writing. Dr. Saad contributed to the formation of the research idea, discussed the experiment design, and the interpretation of experimental results to provide insights and revisions to the manuscript. In the case of Chapter 2, Ph.D. student Omar Mahmoud executed the experiments for the baseline methods.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes two original papers that have been previously published/submitted for publication in peer-reviewed journals/conferences, as follows:

Thesis Chapter	Publication title/full citation	Publication Status
Chapter 2	Kassem, A.;Mahmoud, O. and Saad, S. (2023). Preserving Privacy Through Dememorization: An Unlearning Technique For Mitigating Memorization Risks In Language Models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing	Published
Chapter 3	Kassem, A. and Saad, S. (2023). Finding a Needle in the Adversarial Haystack: A Targeted Paraphrasing Approach For Uncovering Edge Cases with Minimal Distribution Distortion. Submitted to The 18th Conference of the European Chapter of the Association for Computational Linguistics	Under review

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Deep learning models have recently achieved remarkable progress in Natural Language Processing (NLP), specifically in classification, question-answering, and machine translation. However, NLP models face challenges related to security and privacy. security-wise, even small perturbations in the input can significantly impact a model's prediction. This highlights the importance of generating natural adversarial attacks to analyze the weaknesses of NLP models and bolster their robustness through adversarial training (AT). Conversely, Large Language Models (LLMs) are trained on vast amounts of data, which may include sensitive information. If exposed, this poses a risk to personal privacy. LLMs can memorize portions of their training data and reproduce them verbatim when prompted by adversaries. To address these limitations, we delve into the potential of reinforcement learning (RL) based methods in tackling these issues and surmounting the shortcomings present in the existing literature. RL excels in achieving specific objectives guided by a reward function. In pursuit of this, we introduce an End-to-End framework that employs a proximal policy gradient—a reinforcement learning approach—to cultivate a self-learned policy directed by the chosen reward function. The language model (LM) takes on the role of a policy learner. For adversarial attacks, we opt for a combination of the mutual implication score and the negative likelihood of samples generated by the victim classifier. This approach allows us to craft perplexing samples while preserving their semantic significance. In addressing memorization, we employ the negative similarity function, BERTScore, to develop a "Dememorization Privacy Policy." This policy effectively mitigates the risks associated with memorization. Our findings indicate that our framework has proven effective in enhancing the performance of the vanilla classifier by 2% when generating adversarial attacks and reducing LM memorization by 34% to mitigate privacy risks while maintaining the general LM performance.

ACKNOWLEDGEMENTS

I am deeply grateful for the invaluable mentorship and guidance my supervisor, Dr. Saad, provided. His expertise and unwavering support have shaped my academic and professional journey. Under his tutelage, I have expanded my knowledge, developed crucial skills, and gained insights that will undoubtedly serve me well in my future endeavors. Dr. Saad's dedication to my growth as a scholar and his commitment to excellence has been truly transformative, and I am deeply appreciative.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

LM	Language Model
LLM	Large Language Model
NLP	Natural Language Processing
RL	Reinforcement Learning
GPT	Generative Pre-trained Transformer
MLM	Masked Language Model
BERT	Bidirectional Encoder Representations from Transformers
OPT	Open Pre-trained Transformer
COPA	Choice of Plausible Alternatives
ARC	AI2 Reasoning Challenge
Piqa	Physical Interaction: Question Answering
DP	Differntial Privacy
PPO	Proximal Policy Gradient
NLPO	Natural Language Policy Optimization
BLEU	BiLingual Evaluation Understudy
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
PPL	Perplexity
GAN	Generative Adversarial Network.
TPRL	Targeted Paraphrasing via Reinforcement Learning
APT	Adversarial Paraphrasing Task
MSRP	Microsoft Research Paraphrase Corpus
QQP	Quora Question Pairs
SQUAD	The Stanford Question Answering Dataset

CHAPTER 1

Introduction

In recent years, the advancements in Deep Learning models have been awe-inspiring, particularly in Natural Language Processing (NLP). These models have significantly revolutionized various tasks, such as classification, question-answering, and machine translation. However, despite these remarkable achievements, NLP models continue to face critical challenges that deserve attention: security and privacy.

One of the most pressing performance-related challenges is the vulnerability of NLP models to even slight input perturbations. These minor changes in input can cause significant alterations in their predictions, as extensively documented in studies like [10, 12, 1, 19, 11, 5]. Such susceptibility to adversarial attacks can undermine the reliability of these otherwise powerful models.

On a different front, we encounter the rise of Large Language Models (LLMs), trained on vast datasets containing billions of tokens, as seen in [16, 2, 6, 7]. While these models have shown remarkable capabilities, a significant concern arises due to their potential access to sensitive and private information in the training data. The extraction of private data by malicious actors poses a severe threat to personal privacy.

Notably, LLMs have demonstrated the disconcerting ability to memorize portions of their training data and accurately reproduce them when prompted by adversaries [4]. This memorization phenomenon raises profound concerns about data privacy and the need for effective privacy-preserving mechanisms. Addressing this challenge becomes crucial to ensuring the integrity and trustworthiness of intelligent systems.

In the forthcoming sections, we will delve deeper into the issues of adversarial attacks

and data memorization problems, shedding light on some hypotheses that offer explanations for these phenomena. Understanding these challenges and finding robust solutions is essential to unlocking the full potential of NLP models while safeguarding individual privacy and data integrity.

1.1 Textual Adversarial Attacks

Generating adversarial attacks against NLP models is more challenging than for vision models [15]. NLP models rely on discrete word representations, where even slight adjustments can drastically change the meaning or validity of a phrase. Unlike images, NLP models require a deeper understanding of context and language structure, making successful attacks difficult.

Attacks Properties. For a victim classification model, denoted as F_θ, tested on dataset D_t with samples (x_t, y_t), an adversarial attacker aims to perturb x_t to maintain semantic similarity to humans but destroy its meaning when classified by the model. This generates an adversarial example, $x_t^{'}$, that the model misclassifies.

According to the researchers, the source of adversarial behavior in Deep Learning models is an insufficient generalization of the models on unseen data due to a lack of diversity in training samples or a lack of the number of samples necessary to make the model generalize well on unseen data [18]. Another hypothesis is that the linear behavior of Deep Learning models in high-dimensional space leads to the adversarial example problem [8]. Textual adversarial attacks have received less attention than computer vision approaches because the development process of textual adversarial examples has more constraints than computer vision. The text's discrete nature is regarded as the first constraint. In contrast to modifying a few pixels in an image, a slight adjustment of characters or words may be realized, which is regarded as a second constraint. The last constraint is semantic; adding, replacing, or removing a character or word alters the entire meaning of the phrase, as opposed to images, where changing the values of pixels does not affect the overall semantics. Crafting successful adversarial text examples is difficult due to the stated constraints for textual adversarial generation, such as semantic consistency and syntactic legitimacy. In

general, attacks can be noise added to the text to disrupt the model and expose a vulnerability, such as adding many commas or repeated phrases, which may confuse the model and modify its prediction. However, these attacks are frequently unnatural and lack semantic significance. Furthermore, these adversarial examples show the models' unnatural blind spots; as a consequence, the generated instances are not the examples that the classifier is likely to encounter when deployed [8]; even if they exist, they can be easily detected, and eliminated. On the other hand, natural adversarial instances [20] are semantic/syntactic consistent. Furthermore, the generated instances would be similar to those seen by the model during real-world deployment. Textual adversarial attack generation may be divided into four categories: text granularity, model access, target type, and generation approach. Text granularity refers to the level of detail or the size of the textual units within a text. It involves examining how text is divided and organized into smaller components or segments and can be classified as follows:

- Character-level attacks can be generated by inserting, replacing, deleting, or swapping characters in the text. This type may lack semantic/syntactic consistency because it has an easily detectable misspelling or grammatical error.

- Word-level attacks: the attack can be generated using the same actions mentioned in the character level but on the words.

- Sentence-level attacks: the attack can be generated by inserting new sentences or changing the whole structure of the sentence by paraphrasing or style-transfer methods.

Model access refers to the knowledge that can be accessed through the model to initialize the attack; it can be classified into two categories:

- White-box attack: all of the knowledge available about the model can be accessed in this scenario, including but not limited to inputs, outputs, architecture, activation functions, and loss functions. Therefore, white-box attacks can generate a high success attack rate because of the available information about the entity to be attacked.

- Black-box attack: only has access to the output by injecting some examples into the model.

Based on the attack's purpose, it falls into two categories:

- Targeted attack, where the objective is to force the model to produce a specific incorrect prediction
- Untargeted attack, which aims to alter the model's prediction without a specific outcome in mind.

Studies proposed a data augmentation-based approach and an adversarial training-based approach to enhance the model's generalization ability and mitigate the robustness issues. The main idea of data augmentation methods is to augment the training data to add new features that assist the model in generalizing better on unseen data; the methods can use back-translation, paraphrasing, replacing, removing, or adding new words based on the context. One of the drawbacks of data augmentation methods is that it does not guarantee that the generated examples lie in the subset, which confuses the model or which the model fails to classify correctly; in other words, it is not directly designed to generate examples that add new features to the model in order to improve its robustness on the unseen data. Adversarial learning-based approaches solve the previous limitations by generating examples that confuse or classify the model incorrectly to enhance the model's ability on this subset.

1.2 Data Memorization

In the context of Language Models (LMs), data memorization refers to the undesirable behavior where the LM inadvertently reproduces parts of its training data [4]. This phenomenon raises concerns about privacy violations as it exposes sensitive user data. Extensive research [4] has shown that the size of the language model directly influences the degree of memorization. More extensive models tend to exhibit higher levels of memorization, making it crucial to address this issue in developing LMs. Furthermore, when trained

on corpora containing duplicated instances, certain model families demonstrate heightened memorization capabilities. In such cases, the LM becomes remarkably adept at reproducing these duplicated sentences verbatim, exacerbating data exposure's privacy and security concerns [13]. Another factor contributing to memorization is the length of the prompt or context used during LM training or interactions. As the number of tokens in the prompt or context increases, so does the level of memorization. This increase in memorization can be perceived as an adversary gaining access to more information, making targeted attacks more specific, aiming to extract exact information from the LM.

For evaluating memorization in language models, two primary definitions are commonly used. The first one is known as "Eidetic memorization," [3] which identifies a string S as memorized if a prompt P exists, such that the model generates a response Q that includes string S. Both the prompt and the response are present in the training dataset. This definition mainly focuses on measuring verbatim cases of memorization and has found wide usage in the literature [13, 14, 4]. However, a notable drawback of this approach is that even a minor edit in the generated sequence could render it a non-memorized instance, leading to a reduced memorization ratio and creating a false impression of privacy. To address this limitation, the second definition for quantifying memorization is called "approximate memorization," [9], which offers a more accurate way to encompass various types of memorization. In this definition, the model's output for a given prompt p is considered memorized if it falls within a chosen edit distance of the prompt's true continuation present in the training set. This approach allows for a more lenient memorization assessment, capturing cases where the model may produce slightly different but semantically equivalent responses compared to the training data.

Language models memorize data due to their training objective [3]. During the training process, a language model is designed to maximize the likelihood of the data found in its training set, denoted as D. This set comprises various examples, such as specific news articles or webpages from the internet. The primary goal of training is to minimize the loss of predicting the next token accurately, given the previous sequence of tokens, across each training example in the dataset, represented as D. Consequently, this training setup encourages the language model to strive for an "optimal" solution, which involves memorizing

the answer to the question "what token follows the sequence $x_1, ..., x_{i-1}$" for every prefix present in the training set.

In essence, language models become proficient at memorization because they are trained to predict tokens based on contextual information from the data they have been exposed to. As they encounter various sequences repeatedly during training, they develop the capability to recall and generate appropriate tokens, enabling them to mimic coherent and contextually relevant language. However, while memorization may be advantageous during training, it can have drawbacks in real-world scenarios where models need to generalize effectively to previously unseen data. Striking a balance between memorization and generalization is a crucial challenge in developing and deploying robust language models.

1.3 Reinforcement Learning Methods

Reinforcement learning (RL) is a machine learning paradigm focused on training models to make sequences of decisions to maximize cumulative rewards, even when these rewards may not be differentiable. It involves an agent interacting with an environment, taking actions, and learning from their consequences. RL plays a pivotal role in language models in enhancing their performance across various natural language understanding and generation tasks. The critical elements of reinforcement learning pertinent to language models can be briefly outlined as follows:

1. Agent: Within the realm of RL, the agent serves as the decision-maker. The agent is essentially the model itself in language models, such as a neural network-based language model like GPT-3.

2. Environment: The environment represents the external system that the agent engages with. This environment could encompass various applications for language models, including dialogue systems, games, recommendation engines, and more.

3. Actions: In the context of language models, actions correspond to the generation of tokens. The model takes action by producing tokens that directly impact the ongoing dialogue or interaction.

4. Rewards: Rewards are numerical values that the RL agent receives from the environment following actions (in our case, generating tokens). These rewards act as feedback, signaling the agent's performance. In language models, rewards can be determined using metrics or human feedback, such as accuracy, perplexity, ROUGE, and BLEU.

5. Policy: The policy in RL serves as a strategy guiding the agent in action selection based on its current state. For language models, this policy is often embodied by the model's parameters, refined during training to enhance its ability to generate text that maximizes rewards.

6. Learning: RL encompasses the process of the agent learning from its interactions with the environment. Language models facilitate this learning through policy gradient methods like the proximal policy gradient or natural language optimization policy. The model adjusts its policy to elevate the likelihood of choosing actions leading to higher rewards.

7. Exploration vs. Exploitation: In RL, models must balance exploring new actions to discover improved strategies and exploiting known actions that yield high rewards. This trade-off also holds significance in language models, where they must harmonize creativity and coherence in text generation.

1.3.1 Language Models & Policy Gradient

Given tokens $x_{<t} = \{x_0, x_1, \ldots, x_{t-1}\}$ and accumulated hidden states $h_{\theta<t}$ before time step t. An auto-regressive language model (LM) is trained to maximize the probability of the next step token $\hat{x}_t$. LM as a generator G selects the token that has the highest probability x_t as the t-th step decoding output:

$$x_t \sim \text{argmax}_{\hat{x}_t} p(\hat{x}_t | x_{<t}) = \text{G}(x_{<t}, h_{\theta<t}) \tag{1}$$

In the reinforcement learning framework, we define the state at step t as all the sequences generated before t $s_t = x_{<t}$, and the action at step t as the t-th output token

($a_t = x_t$). The policy π_θ represents the probability of selecting token x_t (action a_t) given the preceding state $s_t = x_{<t}$. This probability is derived from the softmax output of the hidden states $\pi_\theta(a_t|s_t) = \text{softmax}(h_{\theta<t})$, and this interpretation extends to the conditional case as well. The single-step reward for token x_t^c at step t can defined as follows:

$$R(x_{c_t}) = E_t \left[\frac{\pi_{\theta_c}(a_t|s_t)}{\pi_\theta(a_t|s_t)} r(x^c{}_t) \right] - \beta \text{KL}(\theta||\theta_c) \tag{2}$$

Where $r(x^c{}_t)$ is the selected reward function. The KL penalty is applied per token using a reference model, which is the original model that does not receive the signal reward to prevent significant deviations.

KL Penalty. The policy of the fine-tuned model may deviate significantly from the old policy (the model before fine-tuning), potentially leading to a less coherent and relevant generation. To address this issue, we introduce a KL divergence penalty term to quantify the dissimilarity between these two policies. This step helps ensure that our optimization process remains within a trustworthy region. The KL divergence, calculated for the policies, is expressed as:

$$KL(\theta||\theta_c) = \sum_{i \in [1,t]} \pi_\theta(a_i|s_i) \cdot \log \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_c}(a_i|s_i)} \tag{3}$$

Policy Gradient. Proximal policy gradient approach with top-p sampling 0.95, known as Natural Language Policy Optimization [17]. Given the reward and the definitions described above, we update our policy at t-th step as:

$$\theta_{\text{new}} = argmax_\theta \mathbb{E} \left[\min \left(r_t(\theta), \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \right) A_t \right] \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The optimization objective is to find the new policy parameters that maximize expected rewards while keeping the policy update bounded within a certain range defined by the clipping parameter. This helps maintain stability during training. PPO also balances the trade-off between exploration and exploitation by encouraging actions that have higher estimated advantages while avoiding drastic policy changes that could disrupt learning.

1.4 Thesis Contribution

The primary objective of this research study was to investigate the impact of RL-based solutions on issues like data memorization and adversarial attacks. Through extensive empirical experiments and analysis, this study has contributed to addressing the literature's limitations. The main contributions of the thesis can be summarized as follows:

- **Advancing Understanding of RL in LM Limitations:** This research significantly advances our understanding of how RL can effectively address LM limitations. For the memorization problem, we demonstrate that employing RL for learning unlearning policies is more effective than likelihood-based methods and pre-processing techniques. This approach effectively tackles the issue without compromising the LM's overall performance in downstream tasks. Moreover, employing RL to generate targeted attacks for LM-based classifiers successfully exposes model vulnerabilities while preserving the semantic meaning of the generated sentences.

- **Empirical Evidence of Framework Effectiveness:** This research provides compelling empirical evidence supporting the effectiveness of the developed framework. We conducted comprehensive experiments across various classification tasks, including sentiment analysis, offensive detection, topic classification, and hate speech detection for an adversarial generation. Our findings reveal that employing adversarial training significantly enhances the performance of the vanilla classifier. Regarding the memorization problem, we applied the framework to nine classification benchmarks and observed minimal to no degradation in the LM's overall performance.

- **Insights into LM Weaknesses:** The findings of this study offer valuable insights for the design and implementation of future work. We identify vital weaknesses of LM-based classifiers, such as their susceptibility to changes in writing style or the introduction of new words, which can confuse the classifier. Additionally, we provide a geometric interpretation by visually comparing the generated and original sentences. Furthermore, the research underscores the importance of providing a practical solution to the memorization problem, as our proposed approach is independent of the

number of protected samples and requires only one training phase, unlike current methods in the literature.

1.5 Thesis Overview

The remaining sections of the thesis are organized as follows: chapter 2 introduces our approach to address the pressing privacy challenges by formulating a ”Dememorization Privacy Policy.” This chapter delves into the core of our methodology and its potential for privacy protection. chapter 3 is dedicated to an in-depth exploration of the adversarial generation challenges, where we unveil our framework for achieving natural adversarial generation. This chapter showcases our dedication to tackling intricate problems. Finally, the thesis concludes with chapter 4, where we summarize our findings, draw insightful conclusions, and propose a forward-looking discussion of potential future research avenues.

References

[1] Moustafa Alzantot et al. “Generating natural language adversarial examples”. In: *arXiv preprint arXiv:1804.07998* (2018).

[2] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[3] Nicholas Carlini et al. “Extracting training data from large language models”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2633–2650.

[4] Nicholas Carlini et al. “Quantifying memorization across neural language models”. In: *arXiv preprint arXiv:2202.07646* (2022).

[5] Yong Cheng, Lu Jiang, and Wolfgang Macherey. “Robust neural machine translation with doubly adversarial inputs”. In: *arXiv preprint arXiv:1906.02443* (2019).

[6] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: *arXiv preprint arXiv:2204.02311* (2022).

[7] William Fedus, Barret Zoph, and Noam Shazeer. *Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*. 2021.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[9] Daphne Ippolito et al. "Preventing Verbatim Memorization in Language Models Gives a False Sense of Privacy". In: *arXiv preprint arXiv:2210.17546* (2022).

[10] Robin Jia and Percy Liang. "Adversarial examples for evaluating reading comprehension systems". In: *arXiv preprint arXiv:1707.07328* (2017).

[11] Robin Jia et al. "Certified robustness to adversarial word substitutions". In: *arXiv preprint arXiv:1909.00986* (2019).

[12] Di Jin et al. "Is bert really robust? a strong baseline for natural language attack on text classification and entailment". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 8018–8025.

[13] Nikhil Kandpal, Eric Wallace, and Colin Raffel. "Deduplicating training data mitigates privacy risks in language models". In: *arXiv preprint arXiv:2202.06539* (2022).

[14] Katherine Lee et al. "Deduplicating training data makes language models better". In: *arXiv preprint arXiv:2107.06499* (2021).

[15] Shilin Qiu et al. "Adversarial attack and defense technologies in natural language processing: A survey". In: *Neurocomputing* 492 (2022), pp. 278–307.

[16] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[17] Rajkumar Ramamurthy et al. "Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization". In: *arXiv preprint arXiv:2210.01241* (2022).

[18] K. Taga, K. Kameyama, and K. Toraichi. "Regularization of hidden layer unit response for neural networks". In: *2003 IEEE Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM 2003) (Cat. No.03CH37490)*. Vol. 1. 2003, 348–351 vol.1. DOI: 10.1109/PACRIM.2003.1235788.

[19] Eric Wallace et al. “Universal adversarial triggers for attacking and analyzing NLP”. In: *arXiv preprint arXiv:1908.07125* (2019).

[20] Zhengli Zhao, Dheeru Dua, and Sameer Singh. *Generating Natural Adversarial Examples*. arXiv:1710.11342 [cs]. Feb. 2018. URL: http://arxiv.org/abs/1710.11342 (visited on 08/19/2022).

CHAPTER 2

Preserving Privacy Through Dememorization: An Unlearning Technique For Mitigating Memorization Risks In Language Models

ALY M. KASSEM, OMAR MAHMOUD, SHERIF SAAD

In Proceedings of The 2023 Conference on Empirical Methods in Natural Language Processing

2.1 Introduction

Large language models (LLMs) have experienced exponential growth in recent years, scaling up from millions to billions to trillions of parameters [33, 10, 13, 16]. As their scale increases, the training sets for these models also expand to billions of tokens [18], leading to overall performance improvements, even in few-shot learning scenarios [10]. However, this growth in model size and training data has raised practical concerns regarding privacy risks associated with memorizing the training data. Adversaries can extract individual sequences from a pre-trained model, even if the training dataset is publicly available [11].

Studies have shown that a language model with 6 billion parameters (GPT-J) can memorize at least 1% of its training data [12]. One potential cause of this memorization is the training strategy of the language model, as its objective is to identify the relationships between tokens, either in an auto-regressive LM setup or through masked language modelling (MLM) [15], where the model predicts the masked tokens based on their surrounding

context [32]. Additionally, repeated instances in the training corpus can contribute to memorization, as more frequent examples are more likely to be memorized [23]. To address the issue of memorization in LLMs, several approaches have been proposed, including data sanitization [26], the application of differential privacy algorithms[1, 3, 24, 40, 4], data deduplication [22], and knowledge unlearning [20]. These techniques aim to prevent the generation of memorized content. However, they also come with certain drawbacks. Data sanitization assumes that private information can be easily identified and is not context-dependent. Differential privacy can lead to lower-quality generative models [3]. On the other hand, knowledge unlearning restricts the number of samples that can be forgotten at once to avoid degrading the overall capability of the language model, which may limit its effectiveness in real-world scenarios.

In this study, we propose a Dememorization framework for mitigating memorization in language models by fine-tuning those models using a efficient reinforcement learning feedback approach with just a few parameter updates.

Given samples of prefixes and suffixes from the original pre-training data of the language model, we use a prefix as input for the language model to generate the suffix; then, we compute the negative BERTScore [44] to measure the dissimilarity between the true suffix and generated suffix, the dissimilarity scores are then regarded as a reward signal to maximize in the training process, which guarantees that the approximate memorization will be mitigated.

We experimented on GPT-Neo & OPT LMs (125M, 1.3B, 2.7B) [7, 43] with reward functions such as BERTScore [44], the weighted sum between perplexity, and SacreBLEU. The dememorization aims to learn a policy π_D that can paraphrase the suffix given some prefix. For example, in**" Alice Green lives at 187 bob street"**, the prefix is**" Alice Green lives at."** The suffix is**" 187 bob street"** We aim that the fine-tuned language model paraphrase the suffix to be: **12 red street**. As the suffix is paraphrased, the memorization relationship between the prefix and suffix is minimized with little to no performance degradation on the initial LM capabilities measured via nine common NLP classification benchmarks (Hellaswag [42], Lambada [29], Winogrande [36], COPA [35], ARC-Easy, ARC-Challenge [14], Piqa[6], MathQA [2], and PubmedQA [21]).

We also evaluate dememorization on increasing the context of the prefix, as many studies show that as a longer context is provided, the memorization ratio increases [11, 12]. The proposed framework does not make any explicit, implicit assumptions or limitations about the data's structure or size to be protected. Also, unlike the DP methods, the proposed framework does not apply any partition mechanism to split the data into public data and private data; as language data cannot be partitioned[9], we apply the policy on all training data as defining, partitioning data into private and public, and limiting the number of samples inadequate in the real-world scenarios.

To summarize, our main findings are the following:

- Using a reinforcement learning feedback approach results in little to no performance degradation of general capabilities while being practical, consistent, & independent of increasing the number of protected samples, and the fluency and coherence of the generated samples are maintained.

- As the language model size increases, the convergence rate improves. Convergence refers to the model-generated suffixes diverging significantly from the original ones while the perplexity difference between generated and original examples decreases. In our experiments, GPT-Neo & OPT 125M converged in four steps with four PPO epochs per batch, GPT-Neo & OPT 1.3B converged in two steps with four PPO epochs per batch, and GPT-Neo & OPT 2.7B also converged in two steps with four PPO epochs per batch.

- As the size of a language model increases, the dissimilarity score increase, which the difference between negative SacreBLEU can measure before and after applying the framework. This suggests that larger models may tend to "forget" the memorized data faster.

- Combining Deduplication with Dememorization enhances privacy with insignificant degradation($\sim$0.5%) in the Language model performance.

2.2 Background

2.2.1 Memorization Definitions

In the context of memorization in large language models, we follow the definition proposed by [23], which introduced approximate memorization. Given a string S, if a prompt P exists, the model generates 's' given 'P,' and the model output is memorized with some chosen edit distance of the prompt's true continuation in the training set. In our study, we choose the edit distance to be a similarity measure (SacreBLEU) as proposed in [19], to be able to capture the approximate memorization, not just the "Eidetic memorization" [11] as the definition of verbatim memorization fails to include more subtle forms of memorization [19].

2.2.2 RL In Language Models

Recently, instruction fine-tuning [45, 28] has emerged as a powerful technique for enhancing LM performance, particularly in reasoning abilities. One approach is to use RL with language models, leveraging feedback from a reward function [34, 45, 28]. This feedback improves the model's performance by maximizing the reward. The reward function can be based on human feedback or automated metrics like BLEU [30] or ROUGE score [25]. While RL has succeeded in various NLP tasks, such as machine translation and question-answering, its application to address memorization in LMs still needs to be explored. In this paper, we investigate using RL with a language model to mitigate privacy risks associated with memorization.

2.3 Related Work

In this section, we delve into recent studies to mitigate memorization in language models, which can be categorized into three main approaches: data pre/post-processing, differential privacy methods, and knowledge unlearning.

Data Pre/Post-Processing: This approach reduces memorization in training data by

applying filters before or after feeding it into the language model. One method is data deduplication [22], which removes duplicates and improves model performance. However, it only partially protects against memorization as the model can still memorize non-duplicate sequences. Another approach is "MemFREE decoding" [19], which efficiently checks the memorization in the LM generation by an n-gram in the training dataset.

Differential Privacy (DP): DP is a widely-used technique for training models to prevent memorization of individual training examples [1]. While effective for fine-tuning language models [41, 24], DP often reduces performance compared to non-private models [3]. State-of-the-art language models are typically trained without DP, using large amounts of data and computational resources. DP algorithms are computationally expensive, slower to converge, and have lower utility compared to non-private methods [3]. Applying DP to language data is challenging due to defining private information boundaries [9].

Knowledge Unlearning (UL): UL[20] is an effective method that reverses the training objective of minimizing the negative log-likelihood for forgotten tokens. It minimally affects language modeling performance in larger models. UL has two approaches: batch unlearning for multiple samples and sequential unlearning for smaller chunks. However, unlearning a large number of samples at once significantly degrades average language model performance. While UL effectively addresses memorization, it has not been tested on sample sizes larger than 128. It does not preserve fluency or coherency for memorized prefixes, which are crucial for practical applications.

In this work, we compare our proposed method with a data-preprocessing approach proposed by [22], which shows that deduplicating helps minimize data memorization. While this method is effective, we show that memorization is still high in the LMs pre-trained with this approach; thus, we show that combining pre-processing with our approach, "Dememorization," effectively mitigates memorization. We also compare our method with UL and show it is not inadequate or impractical in real-world scenarios due to a limited number of samples to forget at once.

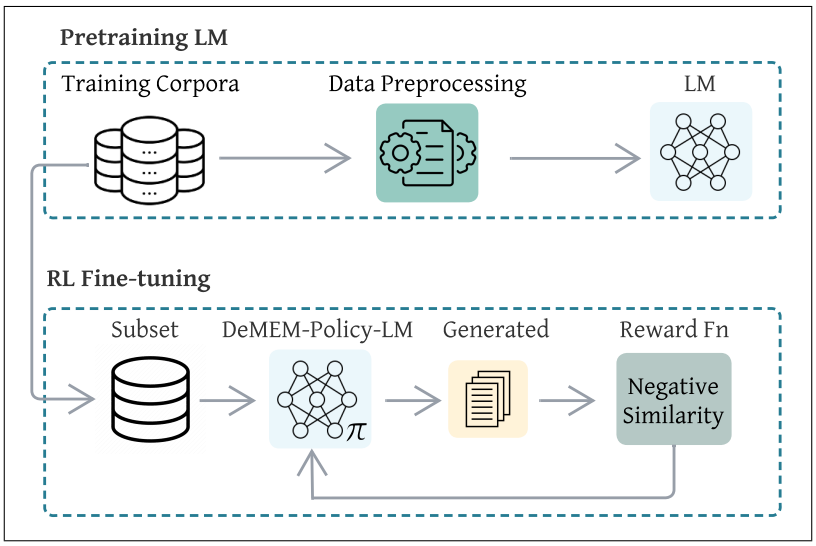


Fig. 2.3.1: First LM is pretrained on a Large Corpora in which Deduplication is Applied. Then Subset of training Corpora is employed to learn the LM a DeMEM Policy Via Negative Similarity Feedback.

2.4 Methodology

2.4.1 Dememorization Via Dissimilarity Policy

We propose an efficient RL feedback loop with the objective of maximizing the dissimilarity between the true & generated suffix (Figure 2.3.1). Using the proximal policy gradient (PPO), we fine-tuned the pre-trained language model in our environment. The environment used is GENERATION AS A TOKEN-LEVEL MDP, similar to the bandit environment in that it presents a random customer prompt and expects a response. The only difference between the employed and bandit environments is that in the bandit, we utilize a discount factor $\gamma = 0.95$ instead of $\gamma = 1$ known as Natural Language Policy Optimization [34](see Appendix A for more details).

Given a dataset with memorized samples, we sample a prefix P and true suffix S_T, the prefix is fed into the pre-trained LM, which generates a suffix S_G; we see how it is dissimilar to the true suffix using BERTScore, and regard that as a reward that encourages the LM to generate dissimilar tokens which minimize the memorization. Each episode starts with a specific prefix which generates the new token in suffix conditioned on this prefix, then by iteratively sampling $\hat{x_{i+1}} \sim f_\theta(x_{i+1}|x_1, ..., x_i)$ and then feeding x_{i+1} back into the model

to sample $x_{i+2} \sim f_\theta(x_{i+2}|x_1, ..., x_{i+1})$.

$$P, S_T \sim D_t \tag{1}$$

$$S_G = f_\theta(s_{G_{i+1}}|x_{P_1}, ..., x_{P_i}) \tag{2}$$

$$Dis_{Score} = -BERTScore(S_G, S_T) \tag{3}$$

This process ends when a certain number of steps have been taken(in our case, the number of steps means some tokens that have been generated) or an end-of-sentence token is generated. The reward for an episode is based on how well the final state(generated suffix) is dissimilar to the target output measured by BERTScore. Furthermore, the KL penalty is applied per token using a reference model (the pre-trained model before fine-tuning.) This prevents the fine-tuned model from generating a suffix that deviates too much from the reference language model (e.g., generating white spaces). A value network V is included beside the language modeling head to estimate the value function. The batch size is 32 for all models; we selected a specific number of steps for each model as the convergence rate for each model is different; we mean by convergence in this context that the model-generated suffixes become significantly different from the original suffixes but without a considerable loss in the perplexity as the difference between the perplexity of the generated examples and original examples becomes smaller, so we selected the appropriate number of steps that balance between these goals.

2.4.2 Measuring Memorization In Language Models

We adopt the concept of approximate memorization, as it provides a more precise and adaptable approach to capturing subtle forms of memorization compared to the limitations of exact memorization. We employ a widely accepted text similarity measure from standard Natural Language Processing (NLP) evaluation techniques to quantify approximate memorization accurately: the **SacreBLEU metric**. SacreBLEU is an improved version of BLEU, known for its stability in measuring the quality of machine-generated text. The BLEU score calculates precision for n-grams ranging from one to four, effectively captur-

ing the degree of overlap between the generated and reference texts. Additionally, a brevity penalty is applied to account for differences in length between the generated and reference texts as follows:

$$\text{BLEU}(\boldsymbol{S_G}, \boldsymbol{S_T}) = \min(1, \tfrac{S_G}{S_T})(\textstyle\prod_{i=1}^{4} \text{precision}_i)^{\frac{1}{4}} \tag{4}$$

To measure forgetting, we consider the negative of SacreBLEU. By utilizing SacreBLEU as a metric for estimating approximate memorization, we define dememorization or forgetting as the process of minimizing the relationship between the given prefix P and suffix S. This relationship represents the information that the adversary seeks to extract based on the given prefix. The metric we mentioned quantifies this relationship. In an example scenario, an adversary has the personal email address **"bob@adam.com"** and seeks to obtain the password. If the LM has memorized this association, it can provide **the password "12345"** when given the email. However, by minimizing or altering their relationship to **"bob@adam.com"** and **the password "0912,"** the LM can generate valid and meaningful output without memorizing sensitive information. This approach achieves the dual objectives of preserving the LM's general capability and the fluency of generated suffixes while ensuring privacy. This solution is more practical in real-world situations than completely removing all information, which can negatively impact the capabilities of the language model (LM).

2.5 Experiments

We assess the effectiveness of the Dememorization approach on forgetting the memorized instances. Next, we evaluate its impact on increasing the context length, as some research suggests that a longer context enhances memorization by providing the adversary with more information. Finally, we evaluate the Dememorized model on downstream tasks to assess its general language model capabilities.

2.5.1 Experimental Settings

2.5.1.1 Memorization Dataset

We employed a subset of the Pile dataset, which was released as a benchmark for training data extraction attacks on large Language Models. Generally, the Pile dataset contains data from 16 different sources (e.g., books, Web scrapes, open source code). We used this version of the subset [1], designed to be easy to extract to assess the performance of targeted attacks. The dataset contains only 15,000 samples since the full version is not released yet. Each sample consists of 200 tokens sampled randomly from the Pile training set. The topics included in the subset are code, news, logs, conversations-replies, copyrights, links, etc. Most of them are in the English language. The dataset is splitted into 13,500 samples for training and 1,500 samples for testing.

Training & Evaluation Data. Each sample consists of a 200-token sequence divided into 100 pre-prefix tokens, 50 prefix tokens, and 50 suffix tokens. During the training phase, we exclusively utilized the prefix and suffix tokens. However, we tested the model in two different settings during the evaluation phase. In the first setting, we evaluated the model's ability to predict the suffix when provided with only the prefix. In the second setting, we evaluated the model's capability to predict the suffix when given the pre-prefix and prefix. This evaluation was designed to assess the model's capacity to protect against acquiring additional information or knowledge. as a longer context in a language model can be considered a form of attack [12].

2.5.1.2 Downstream Tasks

We conduct a thorough evaluation that considers both privacy risks and the expected performance of LMs. This evaluation involves assessing LMs' general capabilities by measuring their performance across diverse classification tasks. The tasks include Hellaswag [42] and Lambada [29] benchmarks, which gauge linguistic reasoning abilities, as well as Winogrande [36] and COPA [35], which measure commonsense reasoning abilities. Addition-

[1] https://github.com/google-research/lm-extraction-benchmark

ally, we utilize ARC-Easy, ARC-Challenge [14], Piqa [6], MathQA [2], and PubmedQA [21] benchmarks to assess scientific reasoning abilities. In addition to these classification tasks. We also measure the perplexity on the Wikitext [27] and Lambada [29] datasets to gain insights into the LMs' language understanding and modeling. Whenever possible, we use the test sets for these evaluations; otherwise, we resort to the validation sets. Also, we did not report Lambada's perplexity & accuracy as it shows so high values for perplexity & low value for accuracy for UL baseline, so to discard the anomaly and better assess the performance.

2.5.1.3 Baseline Methods

For our experiments, we utilized the GPT-NEO family (125M, 1.3B, 2.7B), pre-trained on the publicly available 825GB Pile dataset. Additionally, we employed the OPT family (125M, 1.3B, 2.7B) [43], which was pre-trained on a subset of the deduplicated version of the Pile, along with other corpora from diverse domains. OPT served as our baseline method for deduplication, as per [20], since the deduplicated version of GPT-NEO LMs by [22] were not publicly accessible. We also applied dememorization to the OPT LMs, which can be seen as a combination of the deduplication approach and dememorization, resulting in a significant enhancement in the privacy of these models. Furthermore, we included UL [20] as a second baseline method to highlight weaknesses and distinctions.

2.5.1.4 Implementation Details

For training, we utilized the training subset and fine-tuned the GPT-Neo & OPT LMs fine-tuned them for multiple iterations depending on the model size. To compare our proposed method with UL & deduplication, we followed the configuration proposed by [20] to ensure an adequate comparison, as we randomly sample s samples from the test subset and evaluate the models on those samples for UL since it forgets s samples only at once, we make the LM forget the s samples and then evaluated. To follow the same configuration, we show the average results of 5 random samplings of s samples for all of our experimental settings.

To explore the impact of increasing the sample size to be forgotten, we performed five

random samplings of 32, 128, and 256. Dememorization was carried out using a batch size of 32, and a default value of learning rate of 1.41×10^{-5} was applied to all models. We use the default value of KL Beta of 0.2 and a clip range of 0.2. The GPT-Neo & OPT LMs were employed using the official release in the Hugging Face library. For UL training and memorization evaluation, we utilized the official code provided by the authors, for the selection of hyperparameters see Appendix E. In downstream tasks, we employed the lm-evaluation-harness framework [17] for all baseline methods.

2.5.1.5 Evaluation Metrics

We conducted a comprehensive evaluation of dememorization and baseline methods, employing a multi-perspective approach to assess their effectiveness in three key areas:

(1) Measuring Forgetting: As mentioned in subsection 2.4.2, we employed negative Sacre-BLEU and MA to quantify memorization.

(2) Evaluating Generated Suffixes: To assess text fluency, we utilized the perplexity score of the underlying original model before forgetting. This metric enabled us to assess the grammatical correctness and coherence of the generated suffixes.

(3) Performance on Downstream Tasks: We assessed the performance of the unlearned models across nine classification tasks, employing accuracy scores, and perplexity measurements on Wikitext and Lambada.

2.5.2 Experimental Results & Discussion

We conducted comprehensive experiments to assess the performance of dememorization against the baseline methods. Our main observations are as follows:

2.5.2.1 Overview of The DeMemorization Performance

We comprehensively evaluated the dememorization approach on nine classification tasks, wikitext for perplexity, and the generated samples. The evaluation results, as shown in Table 2.5.1, demonstrate that the dememorization approach effectively provides privacy and decreases the memorization for GPT-NEO while maintaining the LM general capability,

which is measured by evaluating the classification tasks. It also maintains the fluency of the general LM and generated suffix. On the other hand, the UL approach provides more robust protection since it removes the data points completely from the training data, which lowers the general LM capability by a large margin. This is effective privacy-wise but needs to be more practical from the performance perspective. Thus, we tried to balance this tradeoff by employing the dememorization approach..

Model	#Samples	N-SacreBLEU↑	LM(ACC)↑	LM (PPL)↓	GEN(PPL)↓	Epochs/Steps
	32	58.44			3.46	-
NEO_{125M}	128	58.41	43.36	32.28	3.83	-
	256	58.82			3.79	-
	32	99.19	38.62	31098.06	19.77	18
+UL	128	99.69	36.87	9683877.08	6.54	18
	256	99.63	36.34	25146.84	6.03	18
	32	67.07			3.74	
+DeMEM	128	66.21	43.46	33.13	3.93	4
	256	67.05			3.95	
	32	30.76			2.02	-
$NEO_{1.3B}$	128	34.7	48.93	16.16	2.18	-
	256	33.95			2.18	-
	32	99.57	48.61	24.38	4.37	14
+UL	128	98.33	41.55	188.65	5.83	8
	256	99.15	41.34	62.34	5.37	7
	32	52.03			2.44	
+DeMEM	128	51.34	49.40	16.70	2.62	2
	256	52.58			2.65	
	32	26.26			1.8	-
$NEO_{2.7B}$	128	27.25	52.67	13.93	1.92	-
	256	27.37			1.92	-
	32	99.54	49.70	324.68	4.93	11
+UL	128	97.77	47.42	41.50	9.67	8
	256	99.37	39.80	118.68	4.53	8
	32	49.24			2.3	
+DeMEM	128	50.81	52.48	14.15	2.38	2
	256	50.91			2.35	

Table 2.5.1: Main Results: GPT-NEO averaged 5 random samples (s = 32, 128, and 256) for UL. NEO = initial GPT-NEO LM. UL+ = knowledge unlearning, DeMEM = dememorization. LM ACC. = average accuracy of 8 classification datasets, LM PPL = perplexity of Wikitext dataset, GEN PPL = perplexity of generated suffix. Steps for DeMEM & Epochs for UL

2.5.2.2 Deduplication + (DeMemorization & UL)

We included OPT LMs as a baseline for the pre-processing technique, which applies deduplication to decrease memorization. Deduplicating the training data has effectively mitigated memorization, as demonstrated in Tables 1 and 2. OPT models (deduplicated) exhibit higher N-sacreBLEU scores than NEO (non-duplicate version) models while achieving similar or better performance in downstream tasks. However, even in these models, memorization remains high, as only a portion of the memorized samples are duplicates. Therefore, we explored the UL approach and dememorization.

The models that utilized both frameworks benefited significantly and became more robust privacy LMs. While UL reduced memorization by approximately 99% of N-sacreBLEU, it also negatively impacted the general capability of the LM, resulting in an $\sim$11% difference from the original LM across various configurations. On the other hand, dememorization achieved comparable results to UL, with a reduction of $\sim$94% in memorization, without the need to completely remove data points from the training data. In comparison, the loss in general LM capability was insignificant, at around $\sim$0.5%, in the case of 125M and NEO 1.3B dememorization even enhanced performance. These findings suggest that employing a combination of deduplication and dememorization effectively mitigates memorization while maintaining the general capability of the LM. Since data deduplication is applied in most of the recent & large language models [31, 39, 5, 38, 37, 8], we believe our approach combined with deduplication will effectively mitigate memorization.

2.5.2.3 Number of Samples, Stability, & Universal Policy

We investigated the impact of increasing the number of samples on the performance of both UL and dememorization. In line with the findings from [20], UL is sensitive to the number of samples being unlearned simultaneously. Our experimental results in Table 2.5.1, Table 2.5.2 validate this observation. As the number of samples increases, we observe a decrease in the LM's performance. On the other hand, dememorization demonstrates a different behavior as it is unaffected by the number of samples. In dememorization, the LM is fine-tuned one-time using negative similarity as a reward during training, followed

Model	#Samples	N-SacreBLEU↑	LM(ACC)↑	LM (PPL)↓	GEN(PPL)↓	Epochs/Steps
	32	89.24			9.69	-
OPT_{125M}	128	90.98	41.28	31.94	9.76	-
	256	91.03			9.67	-
	32	99.23	37.06	449131.90	12.16	9
+UL	128	99.35	36.48	54917065.46	10.44	9
	256	99.21	37.19	114952.53	13.64	9
	32	94.88			10.86	
+DeMEM	128	95.30	42.25	33.13	10.78	4
	256	95.61			10.58	
	32	71.63			6.72	-
$OPT_{1.3B}$	128	71.96	51.65	16.41	6.92	-
	256	71.7			6.80	-
	32	99.50	39.16	$\star$	11.19	7
+UL	128	99.84	38.67	$\star$	7.93	8
	256	99.52	36.85	$\star$	10.7	7
	32	92.51			9.78	
+DeMEM	128	91.56	51.40	17.39	9.47	2
	256	91.91			9.25	
	32	71.80			6.27	-
$OPT_{2.7B}$	128	67.56	53.74	14.31	6.48	-
	256	66.32			6.3	-
	32	99.15	38.60	$\star$	7.15	11
+UL	128	97.87	41.06	$\star$	13.43	7
	256	99.48	38.20	$\star$	7.6	8
	32	94.53			8.28	
+DeMEM	128	93.08	52.20	15.25	8.31	2
	256	93.24			8.16	

Table 2.5.2: Main Results: OPT averaged 5 random samples (s = 32, 128, and 256) for UL. UL = knowledge unlearning, DeMEM = Dememorization. LM ACC = average accuracy of 8 classification datasets, LM PPL = perplexity of Wikitext dataset, GEN PPL = perplexity of generated suffix. $\star$ means that the value is so high, Reaching infinity. Epochs for UL & Steps for DeMeM.

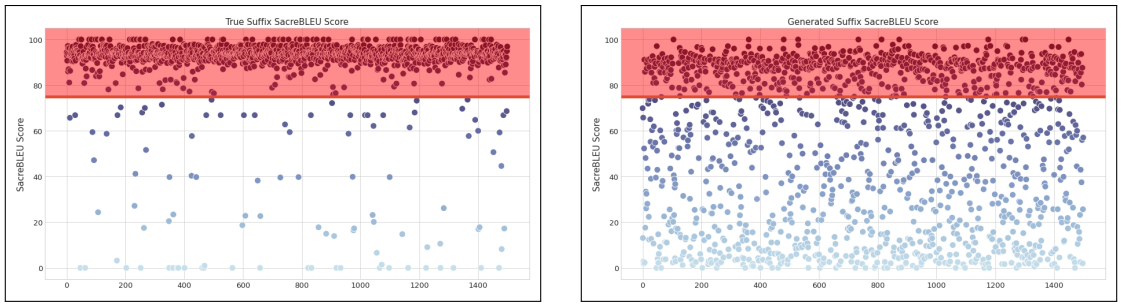
by evaluation on a separate test set. This allows the model to learn a universal policy to forget an unlimited number of samples. Here, the term ”unlimited” signifies the absence of any restrictions, assumptions, or re-training of the LM regarding the number of samples to be unlearned. In UL, however, the model is fine-tuned and evaluated on the same samples to forget them at a time. To unlearn or forget multiple samples, the model needs to undergo fine-tuning multiple times, whether through sequential or batch unlearning. In each iteration, the model is fine-tuned with a specific number of samples (typically 32, as suggested by the authors) to prevent a decrease in the LM's overall capability, this can be regarded as an assumption about a number of samples to be protected at once, which leads to an incomplete solution. See Appendix F for highlighting more assumptions about the UL framework.

2.5.2.4 Perplexity of WikiText & Generated Suffix

Perplexity is a metric used to evaluate the general performance of the LM. We computed perplexity for Wikitext and presented the results in Tables 1 and 2. Dememorization had a minimal impact on perplexity for all models. UL showed significantly higher perplexity in some cases, even reaching infinity. UL's high perplexity is attributed to its gradient ascent approach, which softens the probability distribution and leads to a more uniform distribution and higher perplexity. However, this softening procedure degrades LM performance as the model becomes less confident in generating tokens. We also evaluated the perplexity of unlearned samples, which is crucial in practical applications where the unlearned data domain is used. DeMemorization caused an average degradation of approximately 0.5% in NEO models and around 1.5% in OPT models. UL exhibited higher degradation in both models due to the complete removal of corresponding data points from the model parameters. Dememorized samples can be found in Appendix C.

2.5.2.5 Protection Against Discoverability Phenomenon

Discoverability phenomenon refers to the observation that some memorization only becomes apparent when a model is prompted with a sufficiently long context. [12] found that the fraction of extractable sequences increases in a log-linear fashion with the number of



((a)) Before DeMem ((b)) After DeMem

Fig. 2.5.1: Threshold of 75% SacreBLEU of The Generated Samples Before & After De-Memorization For Neo 2.7B Longer Context

tokens in the context. For example, with a context of 50 tokens, approximately 33% of training sequences can be extracted from the NEO-6B model. However, with a context of 450 tokens, this percentage rises to 65%. We evaluated our dememorization approach by increasing the prefix context from 50 to 150 tokens. The results in Table 2.5.1 show that extending the context does not significantly impact the 125M model in NEO, with a forgetting rate decrease from 58.44% to 45.47%, and has no effect in OPT-125M. However, for larger models like 1.3B and 2.7B, a longer context considerably reduces the forgetting rate by approximately 49% in NEO and around 10% in OPT. Nevertheless, dememorization effectively counters this type of attack, increasing the forgetting rate by approximately 10% for the 125M model and approximately 30% for larger sizes in OPT & NEO as shown in Table 2.5.3. This demonstrates the universality and generalizability of the learned policy across various scenarios.

2.5.2.6 Approximate Memorization Threhold

Based on [19], a BLEU score of 75% for the generated suffix is considered a suitable threshold for determining approximate memorization. However, our investigation found that even a threshold as low as 50% after applying the framework can mitigate this issue. Nevertheless, we chose to use the widely accepted threshold of 75% to demonstrate the effectiveness of our framework. Applying dememorization to the LM resulted in a significant decrease in memorized samples. For GPT-Neo 1.3B and 2.7B, approximate memorization examples decreased from 910 to 497 and 1036 to 321, respectively (refer to Appendix B

Model	BEFORE		AFTER	
	N − SacreBLEU ↑	PPL ↓	N − SacreBLEU ↑	PPL ↓
NEO_{125M}	45.74	4.12	55.04	4.15
$NEO_{1.3B}$	9.11	1.55	36.08	1.71
$NEO_{2.7B}$	10.55	1.41	32.66	1.54
OPT_{125M}	89.35	11.99	94.47	12.38
$OPT_{1.3B}$	59.58	6.64	88.91	7.68
$OPT_{2.7B}$	56.35	5.95	89.37	6.76

Table 2.5.3: Comparsion of Negative SacreBLEU & Perplexity Means Before & After Applying The Framework On a Longer Context; 100 Extra Tokens Combined With The Prefix

for other models). The red region in Figure 2.5.1 represents samples with scores equal to or above 75%. After dememorization, the distribution of samples spreads more evenly across different values instead of being concentrated beyond the 75% threshold. Box plots (see Appendix D) confirm the efficiency of the dememorization approach, as evidenced by the median of the sample's distribution before and after dememorization.

2.6 Conclusion

In this paper, we present a novel framework that tackles the problem of training data memorization in LLMs. We achieve this by employing an RL Dememorization policy. Through extensive evaluations conducted in diverse settings, we demonstrate the effectiveness of our approach. Our framework successfully reduces memorization by significantly decreasing the SacreBLEU score while preserving the overall capabilities of the LM as measured by 9 classification benchmarks.

Limitations

One of the limitations of our work is that it relies on a single scalar reward for optimization, as the problem has dual objectives: dissimilarity and perplexity. To overcome this limitation, we suggest exploring other techniques, such as Multi-objective Reinforcement Learning, which can potentially enhance performance and optimize both objectives simultaneously.

Ethics Statement

Improving the large language model to be privacy-preserving is crucial since the language models have become more prominent and involved in many applications in multi-aspect of life. Ensuring the data privacy of those models is vital since some adversary may be able to reach that information. To make those models widely used, we have to guarantee they cannot emit private data. In this paper, we hope our work will serve as a foundation for developing new and innovative solutions to the problem of approximate memorization in large language models since verbatim memorization can give a false sense of privacy, as earlier work suggested. Our proposed framework provides a promising approach to addressing this issue. Further research and experimentation in this area can lead to even more effective methods for reducing memorization in these models. Our work also highlights the importance of considering both the computational cost and the performance trade-off when developing new techniques for addressing memorization in large language models.

References

[1] Martin Abadi et al. “Deep learning with differential privacy”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[2] Aida Amini et al. “Mathqa: Towards interpretable math word problem solving with operation-based formalisms”. In: *arXiv preprint arXiv:1905.13319* (2019).

[3] Rohan Anil et al. “Large-scale differentially private bert”. In: *arXiv preprint arXiv:2108.01624* (2021).

[4] Priyam Basu et al. “Benchmarking differential privacy and federated learning for bert models”. In: *arXiv preprint arXiv:2106.13973* (2021).

[5] Stella Biderman et al. “Pythia: A suite for analyzing large language models across training and scaling”. In: *arXiv preprint arXiv:2304.01373* (2023).

[6] Yonatan Bisk et al. “Piqa: Reasoning about physical commonsense in natural language”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 7432–7439.

[7] Sid Black et al. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. Version 1.0. If you use this software, please cite it using these metadata. Mar. 2021. DOI: 10.5281/zenodo.5297715. URL: https://doi.org/10.5281/zenodo.5297715.

[8] Sid Black et al. “Gpt-neox-20b: An open-source autoregressive language model”. In: *arXiv preprint arXiv:2204.06745* (2022).

[9] Hannah Brown et al. “What Does it Mean for a Language Model to Preserve Privacy?” In: *arXiv preprint arXiv:2202.05520* (2022).

[10] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[11] Nicholas Carlini et al. “Extracting training data from large language models”. In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2633–2650.

[12] Nicholas Carlini et al. “Quantifying memorization across neural language models”. In: *arXiv preprint arXiv:2202.07646* (2022).

[13] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: *arXiv preprint arXiv:2204.02311* (2022).

[14] Peter Clark et al. “Think you have solved question answering? try arc, the ai2 reasoning challenge”. In: *arXiv preprint arXiv:1803.05457* (2018).

[15] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[16] William Fedus, Barret Zoph, and Noam Shazeer. *Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*. 2021.

[17] Leo Gao et al. “A framework for few-shot language model evaluation”. In: *Version v0. 0.1. Sept* (2021).

[18] Leo Gao et al. “The pile: An 800gb dataset of diverse text for language modeling”. In: *arXiv preprint arXiv:2101.00027* (2020).

[19] Daphne Ippolito et al. “Preventing Verbatim Memorization in Language Models Gives a False Sense of Privacy”. In: *arXiv preprint arXiv:2210.17546* (2022).

[20] Joel Jang et al. “Knowledge Unlearning for Mitigating Privacy Risks in Language Models”. In: *arXiv preprint arXiv:2210.01504* (2022).

[21] Qiao Jin et al. “Pubmedqa: A dataset for biomedical research question answering”. In: *arXiv preprint arXiv:1909.06146* (2019).

[22] Nikhil Kandpal, Eric Wallace, and Colin Raffel. “Deduplicating training data mitigates privacy risks in language models”. In: *arXiv preprint arXiv:2202.06539* (2022).

[23] Katherine Lee et al. “Deduplicating training data makes language models better”. In: *arXiv preprint arXiv:2107.06499* (2021).

[24] Xuechen Li et al. “Large language models can be strong differentially private learners”. In: *arXiv preprint arXiv:2110.05679* (2021).

[25] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[26] Pierre Lison et al. “Anonymisation models for text data: State of the art, challenges and future directions”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 4188–4203.

[27] Stephen Merity et al. “Pointer sentinel mixture models”. In: *arXiv preprint arXiv:1609.07843* (2016).

[28] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[29] Denis Paperno et al. “The LAMBADA dataset: Word prediction requiring a broad discourse context”. In: *arXiv preprint arXiv:1606.06031* (2016).

[30] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[31] Guilherme Penedo et al. “The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only”. In: *arXiv preprint arXiv:2306.01116* (2023).

[32] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[33] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[34] Rajkumar Ramamurthy et al. “Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”. In: *arXiv preprint arXiv:2210.01241* (2022).

[35] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. “Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning.” In: *AAAI spring symposium: logical formalizations of commonsense reasoning*. 2011, pp. 90–95.

[36] Keisuke Sakaguchi et al. “Winogrande: An adversarial winograd schema challenge at scale”. In: *Communications of the ACM* 64.9 (2021), pp. 99–106.

[37] Teven Le Scao et al. “Bloom: A 176b-parameter open-access multilingual language model”. In: *arXiv preprint arXiv:2211.05100* (2022).

[38] Ross Taylor et al. “Galactica: A large language model for science”. In: *arXiv preprint arXiv:2211.09085* (2022).

[39] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[40] Florian Tramèr, Gautam Kamath, and Nicholas Carlini. “Considerations for Differentially Private Learning with Large-Scale Public Pretraining”. In: *arXiv preprint arXiv:2212.06470* (2022).

[41] Da Yu et al. “Differentially private fine-tuning of language models”. In: *arXiv preprint arXiv:2110.06500* (2021).

[42] Rowan Zellers et al. “HellaSwag: Can a machine really finish your sentence?” In: *arXiv preprint arXiv:1905.07830* (2019).

[43] Susan Zhang et al. “Opt: Open pre-trained transformer language models”. In: *arXiv preprint arXiv:2205.01068* (2022).

[44] Tianyi Zhang et al. “Bertscore: Evaluating text generation with bert”. In: *arXiv preprint arXiv:1904.09675* (2019).

[45] Daniel M Ziegler et al. “Fine-tuning language models from human preferences”. In: *arXiv preprint arXiv:1909.08593* (2019).

CHAPTER 3

Finding a Needle in the Adversarial Haystack: A Targeted Paraphrasing Approach For Uncovering Edge Cases with Minimal Distribution Distortion

ALY M. KASSEM, SHERIF SAAD

Under review at The 18th Conference of the European Chapter of the Association for Computational Linguistics

3.1 Introduction

Disclaimer: *This paper contains real-world cases which are offensive/hateful in nature.*

Deep learning models in NLP have made impressive advancements in classification, question-answering, and machine translation. However, they are susceptible to adversarial attacks, which exploit their vulnerability to small input changes [19, 21, 2, 50, 20, 6]. These attacks introduce variations not encountered during training. Two approaches to address these vulnerabilities are data augmentation-based techniques [29, 53, 23, 60] and adversarial training-based approaches [65, 59, 57]. Expanding training data using pre-designed samples generated by data augmentation methods can assist in classifier training. However, generated samples may lack an adversarial nature [1], leading to confusion and inaccurate classification. Adversarial training-based approaches [57, 27, 9, 18, 2] address this limitation by generating challenging examples. This improves the model's ability to handle difficult subsets of data, enhancing robustness and performance.

Generating diverse and semantically meaningful adversarial examples is challenging due to limited operations like word addition, deletion, or substitution. This lack of diversity in word-level generation often results in generated sentences that have identical vectors to the original, offering little insight into the model's behavior.

Recent studies [27, 64] show that character manipulation or word swap methods can produce irrelevant and incoherent samples, altering the original text's meaning. These methods are unsuitable for real-world applications and can harm the model. Attacks lacking semantic significance expose the model's blind spots, are easily detectable and removable, and do not represent real-world examples encountered during deployment.

To address this, researchers have explored sentence-level generation methods. GAN-based approaches [64, 51] show promise in generating diverse examples but can be irrelevant [64]. Incorporating controllable attributes can help mitigate this issue, although obtaining labeled data for some tasks is challenging [51]. Another approach involves paraphrasing using pre-trained language models, which produce diverse examples that fool the classifier but don't target the main weakness for enhancing model performance on the original test set [38]. However, this method lacks targeted fine-tuning to break the classifier and has limited available styles. In this paper, we propose a natural adversarial generation approach through targeted paraphrasing using the FLAN-T5 language model [8] as a generator and a classifier task as a discriminator. We train our generator with a self-learned policy via proximal policy gradient (PPO), a reinforcement learning technique [44]. Our framework consists of two steps: training a paraphrasing model using FLAN-T5 (seq2seq) on seven datasets, carefully filtered to ensure maximum diversity in the generated paraphrases. Then, we fine-tune the paraphraser using RL, taking into account the reward function based on the classifier's confusion and mutual implication scores. This ensures that the meaning of the text is preserved while generating adversarial examples. We evaluated our proposed framework, Targeted Paraphrasing via RL (TPRL), on four classification tasks: sentiment analysis, news classification, hate speech, and offensive speech. We conducted both automatic and human evaluations. Our experiments showed that TPRL could generate adversarial examples that enhance the classifier's performance. Incorporating challenging adversarial samples during training makes the classifier more robust and powerful. Our

findings can be summarized as follows:

- Utilizing the generated examples for adversarial training improves classifier performance on original and adversarial test sets.
- Our work demonstrates that the learned policy for one classifier is universal and can generalize to unseen classifiers in the same dataset.
- Experiments show that TPRL outperforms strong baselines and improves results on various models and datasets.

3.2 Background

This section briefly introduces and formalizes textual adversarial attacks for text classification and employs RL in language models for generating adversarial attacks and other tasks.

3.2.1 Textual Adversarial Attacks

Generating adversarial attacks against NLP models is more challenging than for vision models [39]. NLP models rely on discrete word representations, where even slight adjustments can drastically change the meaning or validity of a phrase. Unlike images, NLP models require a deeper understanding of context and language structure, making successful attacks difficult.

Attacks Properties. For a victim classification model, denoted as F_θ, tested on dataset D_t with samples (x_t, y_t), an adversarial attacker aims to perturb x_t to maintain semantic similarity to humans but destroy its meaning when classified by the model. This generates an adversarial example, $x_t^{'}$, that the model misclassifies.

3.2.2 RL In Language Models

Reinforcement Learning (RL) is a powerful technique for training machine learning models to learn complex tasks. RL has gained interest in enhancing performance, particularly in natural language processing (NLP) tasks using Transformer-based models [49, 10, 40, 4]. Language models like Transformer-based models have excelled in NLP tasks such as machine translation, language generation, and question-answering. RL can be applied to language models through fine-tuning, using a reward function to guide policy learning [42]. The reward function, based on human feedback [66, 35] or an automatic metric like BLEU [36] or ROUGE score [28], provides feedback to improve the model's performance. Despite RL's success in NLP tasks, its potential for generating adversarial attacks remains largely unexplored. This paper presents the first investigation of using RL with a language model for generating natural adversarial attacks.

3.3 Collecting Labeled Paraphrasing Pairs

This section will review the selected datasets for training the paraphrase model. Afterward, we will outline a systematic procedure for filtering those datasets to maximize the paraphrase pairs' diversity, similarity, and relevance.

3.3.1 Paraphrasing Datasets

The first stage of our approach involves collecting seven diverse paraphrase datasets. Most of the datasets undergo meticulous human judgment annotation, ensuring our models are trained on high-quality paraphrasing examples.

We have developed a comprehensive and varied paraphrasing corpus, compiled from seven distinct paraphrasing datasets, namely the APT dataset [34], Microsoft Research Paraphrase Corpus (MSRP) [11], The PARANMT-50M corpus [55], TwitterPPDB [25], PIT-2015 [58], PARADE [15], and QQP (Quora Question Pairs) [17]. To ensure optimal quality, we employed the filtered version of the PARANMT-50M corpus, as suggested by [24]. Furthermore, we retained only sentence pairs with similarity labels of 4, 5, and 6 from TwitterPPDB, thereby ensuring high relevance and diversity. Similarly, we only incorporated sentences with semantic similarity labels of 5 and 4 from PIT-2015. Finally, we selected samples labeled as duplicates from QQP. The merged dataset comprises 560,550 samples, which we subjected to a filtering procedure to promote high-quality similarity and diversity in the subsequent stage.

3.3.2 Improving Diversity & Relevance Via Data Filtering

In the second stage of our approach, we select appropriate training data for the paraphrase model using inspiration from [24]. Despite the availability of human annotations, it is still possible for noise and irrelevant samples to exist in the dataset. We employ aggressive filtering with four filters to address noise and irrelevant samples in the dataset. Firstly, we remove sentence pairs with over 50% unigram overlap, ensuring lexical diversity computed using SQUAD evaluation scripts based on the F1 score [41]. Secondly, we discard pairs with less than 50% reordering of shared words, promoting syntactic diversity measured

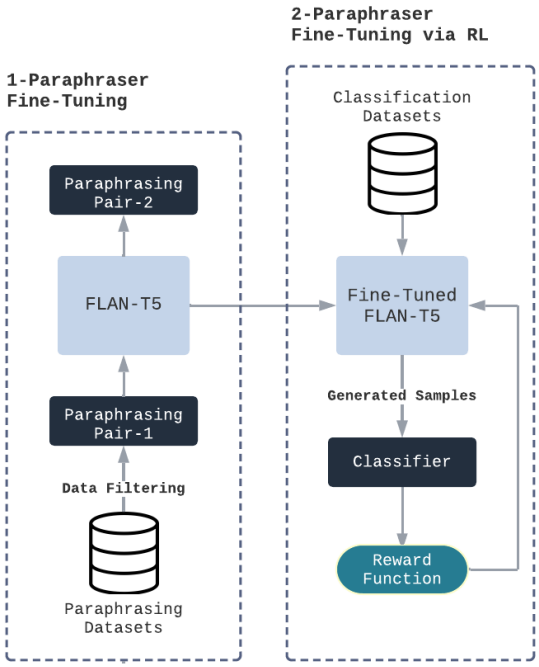


Fig. 3.4.1: Components of our framework **TPRL** for Natural Adversarial Generation. (1) Employing Data filtering and then paraphraser fine-tuning. (2) Targetted paraphrasing through employing RL on classification Datasets.

by Kendall's tau [22]. Thirdly, we eliminate pairs with less than 50% semantic similarity, measured by cosine similarity using the "all-MiniLM-L12-v2" model [52, 43]. Finally, we remove sentences with over 70% trigram overlap to improve diversity further. After applying these filters, the refined dataset contains 96,073 samples, split into training (76,857 samples), validation (9,608 samples), and testing (9,608 samples) sets. These filters ensure a diverse and representative sample for effective training and evaluation.

3.4 Targeted-Paraphrasing Via RL

Figure 3.4.1 illustrates the framework's structure. After filtering data for diverse and relevant paraphrase pairs, we proceed with the initial fine-tuning of the model. Subsequently, we utilize the proximal policy gradient, a reinforcement learning technique, to further fine-tune the model. This approach generates paraphrases that exploit the classifier's weaknesses, resulting in complex and effective adversarial samples.

3.4.1 Paraphraser Model

We fine-tune the FLAN-5-large language model, a variation of T5, on over 1000 additional tasks using chain-of-thought data. This instruction fine-tuning improves performance on various tasks. The model is fine-tuned for nine epochs as more epochs showed to make the model over-fitting, with the first paraphrase pair going through the encoder and the second through the decoder. Using the BERT-Score metric [62], the model achieves an F1-score of 75.925%. Pre-training the LM enhances output fluency, diversity, relevance, and paraphrasing capability. It also addresses task-irrelevant generation in sentence-based attack methods by employing LM and data filtering techniques. The LM is trained on relevant pairs, maximizing task-specific outputs. Utilizing an LM enhances paraphrasing capability, as LMs serve as a knowledge base with not only linguistic knowledge but also relational knowledge present in pre-training data[37]. This knowledge improves generation quality by introducing new information about entities or objects in the input text.

3.4.2 Fine-Tuning Paraphraser Via RL

After the initial fine-tuning, the paraphraser model can generate various relevant and fluent paraphrases. To enhance its performance, our approach includes a guiding component. This involves further fine-tuning the model using reinforcement learning (PPO), enabling it to produce targeted adversarial examples that confuse the classifier.

RL Feedback Loop. The language generation process is a discrete sequence of actions where the generator, represented by G, interacts with the classifier, represented by C, which acts as an environment. We use a proximal policy gradient approach with top-p sampling 0.95, known as Natural Language Policy Optimization [42](see Appendix A for more details).

To maximize the reward r_t returned from the environment, we start with an original example (x, y) from dataset D and choose a vocabulary word as an action a_t based on the current input and previous actions as follows:

$$\hat{a_t} \sim p_G(a_t|x, \theta) \tag{1}$$

where θ represents the parameter of the generator. Each episode starts with a specific prompt x and ends when a certain number of steps have been taken, defined as the maximum length of the original sentence to prevent the generator from adding irrelevant tokens. Additionally, we use early stopping to prevent the model from generating meaningless or irrelevant tokens. With these constraints, we allow the generator to add new, relevant information or remove and replace existing tokens, giving it complete freedom to sample diverse, relevant, and new tokens from the action space.

After sampling the action sequences $\hat{a}_t$, we get a new example $(x^{'}, y)$ through generation. We then feed this generated example $(x^{'}, y)$ into the classifier to obtain the reward $r(\hat{a}_t)$, which is defined as a weighted sum of confusion $(1 - confidence)$ and Mutual Implication as follows:

$$r(\hat{a}_t) = \beta * (1 - p_C(y|\hat{x}; \psi) + \alpha * MI(x, \hat{x}) \tag{2}$$

Where ψ represents the parameter of the classifier. β & α are the weighting factors, we set them to 0.5

The first term encourages the generator to learn a policy that generates confused samples, while the second term ensures the similarity & naturalness of meaning and semantic equivalence, which evaluates the degree to which the paraphrases imply the same meaning as the original text. We estimate the Mutual Implication using ALBERT XXLarge v2 [26] trained on various datasets [32, 3, 56, 32, 33]. The KL penalty is applied per token using a reference model to prevent significant deviations. In the RL feedback loop, we generate ten alternative samples and assess their adequacy and fluency using the Parrot Tool[1] [2], and select the best one for reward computation to maximize the fluency and relevance. The model was trained for thirty epochs with a batch size of 32 and optimized using the Lion optimizer [5], using a learning rate of 4.9×10^{-6}; see Appendix K for more details about the hyperparameters.

[1] https://huggingface.co/prithivida/parrot_adequacy_model
[2] https://huggingface.co/prithivida/parrot_fluency_model

3.4.3 Adversarial Training

After fine-tuning the paraphraser with RL feedback, we use its improved performance to generate adversarial samples from the original training set. These samples are added to the training set, creating an updated dataset. Using the same random seed, we train a new classifier from scratch, employing adversarial training.

We generate counterparts for each sample in the original training set to create adversarial samples. However, not all generated counterparts are useful for training, as discussed by [57], because we specifically target the edge cases that can improve classifier performance. To filter out irrelevant counterparts, we exclude those with a mutual implication score below 50%, ensuring that only semantically equivalent samples are included in the updated dataset. The number of generated samples depends on the dataset and classifier used.

For the classifiers, we began by fine-tuning the classification models (subsubsection 3.5.1.2) on the respective dataset (subsubsection 3.5.1.1) and reported their performance. Each model achieved a different performance and made different errors. We then leveraged these classifiers within the RL feedback loop.

3.5 Experiments

We assess the effectiveness of Targeted-Paraphrasing Via RL adversarial attacks (TPRL) on four distinct classification tasks: sentiment analysis, news topic classification, hate speech detection, and offensive speech detection. To this end, we carefully select relevant datasets, outline our implementation details, establish baseline methods, and specify evaluation metrics.

3.5.1 Experimental Settings

3.5.1.1 Tasks & Datasets

Sentiment Analysis. *SST-2 & SST-5* datasets for sentiment analysis in movie reviews from the Stanford Sentiment Treebank [45]. SST-2 (N=6920)has binary sentiment labels (positive or negative), while SST-5(N=8540) has more fine-grained sentiment labels (very

positive, positive, neutral, negative, and very negative) with an average of 19 words per sample.

New Topic Classification. *AG News* dataset [63] with a number of samples 120,000, categorizing news articles into four classes: World, Sports, Business, and Science/Technology with an average of 38 words per sample.

Offensive Speech Detection. *SemEval2019 Task 6 (OffensEval)* dataset (N=11916)[61] for offensive detection in tweets has binary classes: offensive and non-offensive tweets with an average of 19 words per sample.

Hate Speech Detection. *Hate speech* dataset(N=7071), a collection of sentences extracted from Stormfront, a white supremacist forum [12]. Based on their content, sentences are categorized into two different classes: HATE and NoHate, with an average of 16 words per sample.

We eliminated all punctuation, mentions, hashtags, and URL links from the samples of all datasets. Furthermore, we employed lowercase for all samples. The maximum sequence length was implemented as the maximum length parameter in all BERT models. The training, validation, and test sets officially released by the creator of the datasets were utilized.

3.5.1.2 Victim Models

To assess the effectiveness of our approach across different models, we selected five popular pre-trained language models: *BERT-base*, *BERT-large* [10], *RoBERTa-base*, RoBERTa-large [30], and DeBERTa-v3-large [14], which vary in architecture and size.

3.5.1.3 Baseline Methods

TTPRL is compared to SCPN [18] and StyleAdv [38], other sentence-level adversarial attack techniques. SCPN uses a seq2seq Bi-directional LSTM [16] with the PARANMT-50M corpus and syntactic templates. StyleAdv utilizes the STRAP model [24] for style transfer, incorporating five distinct styles. Also, we considered an untargeted paraphrasing model (UNTP) without the guiding component. However, these methods have limitations in diversity and robustness of adversarial sample generation. We used official code or

Classifier	SST2	SST5	AG's News	HS‡	OFF‡
BERT$_{Base}$	90.88	53.52	93.97	⋆	84.76
+*SCPN*	89.67	51.71	93.26	⋆	83.60
+*StyAdv*	87.91	52.35	93.19	⋆	81.86
+*UNTP*	⋆	51.90	94.17	⋆	⋆
+TPRL	**91.15**	52.39	**94.46**	⋆	**85.11**
BERT$_{Large}$	91.43	53.52	94.18	⋆	85.11
+*SCPN*	90.66	53.61	93.17	⋆	72.09
+*StyAdv*	90.17	23.07	93.57	⋆	72.09
+*UNTP*	⋆	54.84	**94.42**	⋆	⋆
+TPRL	**92.58**	**54.93**	94.36		**85.58**
RoBERTa$_{Base}$	94.34	54.79	93.78	91.90	83.95
+*SCPN*	92.31	53.89	93.61	91.30	82.67
+*StyAdv*	91.81	52.21	93.32	91.55	82.32
+*UNTP*	⋆	**56.19**	93.78	91.90	⋆
+TPRL	94.00	56.15	**93.93**	**92.45**	**85.00**
RoBERTa$_{Large}$	93.73	58.30	93.92	92.45	85.93
+*SCPN*	49.91	23.07	93.80	92.30	72.093.
+*StyAdv*	49.91	23.07	93.73	91.75	72.09
+*UNTP*	⋆	⋆	93.86	92.45	⋆
+TPRL	**94.72**	**58.95**	**94.21**	92.05	**84.53**
DeBERTa-V3$_{Large}$	94.89	58.46	93.92	89.50	84.65
+*SCPN*	93.52	**59.23**	93.75	89.50	84.76
+*StyAdv*	93.41	55.42	⋆	**92.95**	80.93
+*UNTP*	⋆	58.09	93.85	89.50	⋆
+TPRL	**95.82**	58.77	**94.22**	92.30	**85.93**

Table 3.5.1: The Classifier-Dataset Experiments For Five Classifiers. We Show The Accuracy Results On The Original Test Set Before & After We Apply AT With TPRL & The Three Baseline Methods. ‡ Refer to Hate Dataset, OFF to Offensive Dataset. * Refer to not deployed experiments. The best comparable performances are bolded

followed the same methodology and hyperparameters for the baselines (see Appendix H for implementation details). UNTP failed to meet the criteria for the SST2, Hate speech, and offensive speech datasets. Generator collapse occurred during training for BERT$_{Base}$ in the hate speech dataset. For a fair comparison, AG's News with DeBERTa-V3$_{Large}$ was excluded due to low accuracy in adversarial training.

Classifier/Framework	SCPN (%)	StyleAdv (%)	UNTP (%)	TPRL (%)
BERT$_{Base}$	69.37	30.62	64.43	34.84
+AT	68.70	54.28	66.24	42.28
BERT$_{Large}$	70.86	29.97	64.89	41.51
+AT	68.00	51.73	64.35	44.36
RoBERTa$_{Base}$	95.53	29.29	67.52	37.27
+AT	74.01	57.42	67.27	43.41
RoBERTa$_{Large}$	82.49	28.02	89.78	40.27
+AT	75.55	52.28	90.19	49.22
DeBERTa-v3$_{Large}$	74.77	36.79	63.38	34.66
+AT	75.01	45.33	64.83	47.82

Table 3.5.2: The Classifier-Framework Experiments For Five Classifiers. We Show The Accuracy Results On The Adversarial Test Set Before & After We Apply AT With TPRL & The Three Baseline Methods..

3.5.2 Evaluation Metrics

We thoroughly evaluated TPRL's effectiveness in four key areas, ensuring the following:

(1) Improving Performance: We evaluated TPRL's impact on the accuracy of the original test set, alone and in combination with other methods, to assess its overall performance enhancement.

(2) Fluency and Quality: We assessed fluency using perplexity (PPL) from GPT-2-XL [40] and a RoBERTa-large classifier trained on the CoLA corpus [54], to overcome the limitations of perplexity in evaluating fluency as the model provides accurate grammatical acceptability judgments.

(3) **Semantic Similarity (SIM):** We assessed semantic similarity between input sentence and generated samples using the "all-MPNet-Base-v2" embedding-based SIM model [46, 43], known for its performance on semantic textual similarity (STS) benchmark [31]. We also used the mutual implication (MI) metric to capture inferential semantics comprehensively, addressing the limitations of STS.

(4) **Validity:** We conducted human evaluations to determine the percentage of samples that produced adversarial examples without altering the original label. To overcome human evaluation cost, we employed ChatGPT, which has demonstrated comparable or superior performance to crowd-workers in text annotation tasks [13, 47, 7]. This assessment validated the credibility of TPRL-generated samples.

Classifier	Framework	PPL ↓	FL ↑ (%)	SIM ↑ (%)	MI ↑ (%)
BERT$_{Base}$	SCPN	567.66	50.57	74.49	80.58
	StyleAdv	670.47	57.65	74.62	58.34
	UNTP	498.61	85.20	**77.33**	**92.41**
	TPRL	**368.41**	**87.1**	73.56	89.9
BERT$_{Large}$	SCPN	565.61	50.71	74.67	80.94
	StyleAdv	863.50	56.42	74.14	57.40
	UNTP	373.52	85.97	**77.64**	**91.25**
	TPRL	**372.51**	**86.84**	73.88	89.82
RoBERTa$_{Base}$	SCPN	563.46	50.31	74.43	80.28
	StyleAdv	708.28	57.26	75.66	58.99
	UNTP	**254.37**	83.31	**78.32**	89.92
	TPRL	492.52	**85.58**	71.22	**89.99**
RoBERTa$_{Large}$	SCPN	555.02	50.43	74.43	80.19
	StyleAdv	579.80	57.34	74.30	56.81
	UNTP	**230.73**	83.32	78.44	88.49
	TPRL	302.76	**87.38**	**70.84**	**90.61**
DeBERTa-v3$_{Large}$	SCPN	560.70	50.29	74.28	79.91
	StyleAdv	845.76	57.69	**74.92**	57.46
	UNTP	**372.44**	73.21	68.86	73.14
	TPRL	393.06	**87.27**	73.90	**89.67**

Table 3.5.3: Automatic Evaluation Results Showing The Average of Generated Adversarial Training Samples of The Five Datasets Across Selected Classifiers & Baselines. The best comparable performances are bolded

3.5.3 Experimental Results & Discussion

We conducted comprehensive experiments to answer the following three overarching questions regarding TPRL:

3.5.3.1 Does TPRL Enhance The Performance?

We evaluated TPRL on multiple datasets, tasks, and victim models, comparing it to three baseline methods. The results in Table 3.5.1 consistently demonstrate that TPRL improves performance on the original test set across diverse datasets and victim models. TPRL also excels in fluency and relevance, essential factors for an effective adversarial generation. In contrast, SCPN, StyleAdv, and UNTP did not consistently enhance performance on the test set across various classifiers and datasets. This inconsistency is due to the absence of a guiding component specifically targeting classifier weaknesses. While UNTP outperformed SCPN and StyleAdv in performance enhancement, it still lacked consistent effectiveness. TPRL's generated sample count varied by dataset and classifier, with no correlation between the number of samples and improvement gained after Adversarial training. TPRL focuses on targeting classifier weaknesses rather than simply changing labels. TPRL's average number of generated samples for each dataset on the five classifiers are as follows: SST-2: 359, SST-5: 2993, HS: 135, OFF: 407, and AG's News: 3825. In terms of the adversarial test set, SCPN did not enhance classifier performance, StyleAdv showed significant improvement but not consistently on the original test set, and UNTP had a small and sometimes non-existent improvement. TPRL achieved an $\sim$8% improvement on the adversarial test set while maintaining a balance in the tradeoff between the performance on both the original and adversarial sets.

3.5.3.2 Does TPRL Generate Relevant Samples?

In an adversarial generation, maintaining topic and meaning while fooling the classifier is crucial since a sample can easily change the classifier's decision if the meaning changes. This concern is even more significant for sentence-level methods that create new sentences. We evaluated the generated samples using three approaches: (1) Automatic evaluation for similarity and fluency, including PPL for accurate fluency assessment. (2) Human evaluation for validity. (3) Visualization techniques for observing the geometric interpretation of samples. Table 3.5.2 shows the results, with TPRL and UNTP achieving the lowest PPL and superior generation quality. TPRL also outperforms other baselines in fluency, ranging

from 86% to 87%, across classifiers and datasets, thanks to including the MI score in the reward function to encourage natural sentence generation.

Original: i complain all the time about seeing the same ideas repeated in films over and over again , but the bourne identity proves that a fresh take is always possible.
Adversarial: ive always been complaining that movies repeat the same ideas but the identity of bourne proves that you dont need to reinvent the wheel
Original: something must be done to halt this terrible wave of immigrants entering britian
Adversarial: in order to stop a terrible wave of immigration into britian it is necessary to do something
Original: i live in oslo and race mixing is a huge problem
Adversarial: theres a huge problem of race mixing in oslo
Original: william shatner as a pompous professor is the sole bright spot
Adversarial: the only bright spot of the movie is william shatner in a pompous professor
Original: if you enjoy more thoughtful comedies with interesting conflicted characters; this one is for you
Adversarial: this is a good comedy for those who like thoughtful comedies with interesting characters
Original: any ways smearing a single dna steam with black dna is horrible
Adversarial: any way it seems disgusting to smear one steam with black dna
Original: we spend so much time on black people when the real problem is asians and hispanics
Adversarial: we spend too much time and effort worrying about black while the asians and hispanics were the ones

Table 3.5.4: Comparing Original & Adversarial Examples Generated by TPRL.

Regarding The Relevance: TPRL surpasses baseline methods in MI. For specific configurations, cosine similarity scores are low ($\sim$74%), while MI scores are high ($\sim$89%). This discrepancy arises because cosine similarity struggles to capture the inferential role accurately (further explanation given in section 3.6). See Appendix J for dataset-specific results.

For Human evaluation: Following [38], considering the cost, we conducted a validity evaluation on SST2. We randomly selected 100 adversarial samples for TPRL, SCPN, and StyleAdv (36, 33, and 31 samples, respectively). Each sample was evaluated by three an-

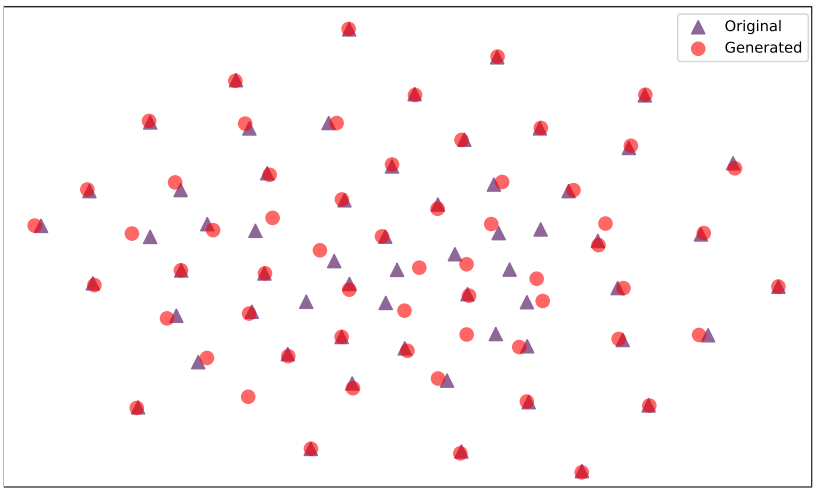


Fig. 3.5.1: T-SNE visualization of the vectorized original and TPRL-adversarial sentences in the SST-2. The adversarial sentences (circles) mostly overlap with the original sentences (triangles), suggesting that generated sentences maintain the original class distribution.

notators who determined if the sentiment matched the original example. The final decision was made by voting. The percentage of valid adversarial samples was: TPRL 72%, SCPN 51.5%, and StyleAdv 32.2%. TPRL achieved the highest validity, confirming minimal distortion to the original distribution. To evaluate the similarity of larger generated samples, we employed ChatGPT (GPT-3.5). Randomly choosing 100 samples from each framework in the SST-2 dataset, we rated them on a scale of 1 to 5, where 1 indicated significant dissimilarity, and 5 denoted substantial similarity. TPRL exhibited superior performance to the three baselines, receiving the highest similarity ratings across categories 5, 4, and 3. Further details can be found in Appendix L.

For Visualization: We randomly selected 60 samples from the SST-2 dataset of the same class and transformed them into vectors using Sentence-BERT ("all-MPNet-Base-v2" model). T-SNE [48] was used to generate a 2D representation of the vectorized samples (see Figure 3.5.1). TPRL-generated samples closely resemble the original data, overlapping or partially overlapping with the original sentences. This observation highlights that even a slight movement in the semantic space can deceive the classifier, exposing LM-based classifiers' vulnerabilities. We obtained similar outcomes consistently across various datasets, classifiers, and random samples.

Policy/Classifier	**BERT**$_{Base}$ (%)	**BERT**$_{Large}$ (%)	**RoBERTa**$_{Base}$ (%)	**RoBERTa**$_{Large}$ (%)	**DeBERTa-v3**$_{Large}$ (%)
None	90.88	91.43	94.34	93.73	94.89
Policy-BERT$_{Base}$	91.15	92.09	93.35	$\star$	95.38
Policy-BERT$_{Large}$	91.04	92.58	**94.45**	94.94	**96.15**
Policy-RoBERTa$_{Base}$	90.06	**93.24**	94.0	$\star$	95.60
Policy-RoBERTa$_{Large}$	**91.87**	92.36	$\star$	94.72	95.60
Policy-DeBERTa-v3$_{Large}$	90.93	**93.24**	94.12	**95.49**	95.82

Table 3.5.5: Accuracy Results of Different Classifiers Trained With The Examples Generated By Various Attacking Policies On The SST-2 Dataset. Showing the Universal Policy. The best comparable performances policy for the classifier is bolded

3.5.3.3 Does TPRL Learned Attacking Policy Universal?

To investigate this question, we employed a fine-tuned generator to target specific classifiers, such as BERT-Base. The generated samples were then utilized to fine-tune another classifier, for instance, BERT-Large, to observe the impact on performance. Since we had five classifiers, we employed the generated samples from one classifier to fine-tune the remaining four. Table 3.5.5 displays the outcomes in SST-2 dataset(see Appendix I for the remaining datasets), revealing that most classifiers benefited from the samples generated by other classifiers, surpassing the naive baseline. This underscores the universality of the learned attacking policy. Notably, in certain instances, the improvement achieved for the attacked classifier equaled that of the transferred classifiers despite each classifier having distinct errors prior to adversarial training(Appendix M). This can be interpreted as differences in size, architecture modifications, and training data; all the classifiers share a common architecture based on Transformers models, and even the errors are different, but the policy targets the universal weakness. The universal policy holds across most configurations, encompassing datasets and classifiers, demonstrating that the learned attacking policy possesses both universal and model-specific features.

3.6 Ablation studies

In this section, we perform ablations on TPRL to understand its key components' impact on improvements over baselines. We also validate the accuracy of MI's similarity measurement and distinguish it from cosine similarity.

TRPL Diversity Enhances Classifier Performance: We investigate the role of diversity in performance improvement. Through qualitative inspections, we compare TPRL and baseline methods that lack diverse dataset training & data filtering. Interestingly, we observed that higher SIM (Similarity) scores correlate with reduced diversity and classifier performance after adversarial training. In contrast, the MI (Mutual Information) metric maintains diversity in generations, explaining the higher SIM scores but lower MI and Accuracy metrics of baseline methods since they don't generate diverse samples. Human evaluations confirm the alignment of MI with their assessments. Furthermore, we confirm our observation by utilizing ChatGPT (GPT-3.5) to assess the diversity of generated samples for the three baselines as TPRL outperformed the three; see Appendix L for more details.

The Importance of Targeted Component: Our TPRL implementation has a targeted component using RL. However, we evaluated using only the fine-tuned paraphraser without RL (UNTP) showed inconsistent performance compared to TPRL (Table 3.5.1). Similarly, other baselines requiring a targeted component also yielded minor improvements. TPRL outperformed other models across most metrics (Table 3.5.3), emphasizing the importance of the targeted component.

3.7 Conclusion

In this work, we introduce TPRL, a novel adversarial generation approach aimed at enhancing the robustness of classification models. Our key innovation lies in incorporating a targeted component through reinforcement training, enabling the automatic acquisition of various attacking policies. We validate the effectiveness of TPRL across four distinct classification tasks, where our experiments consistently demonstrate a superior generation of natural adversarial samples. Remarkably, these samples accurately represent edge cases

while exhibiting minimal distortion in the underlying data distribution.

Limitations

One limitation of our work is the reliance on a single scalar reward for optimization, despite the problem having dual objectives: confusion and maintaining similarity. We recommend investigating alternative techniques, such as Multi-objective Reinforcement Learning, to address this limitation. This approach has the potential to enhance performance by optimizing both objectives concurrently. Moreover, the datasets used in paraphrasing currently need longer sequences, approximately 256 tokens, which restricts our approach to generating adversarial samples for longer sequences.

Ethics Statement

Enhancing classifier performance is of utmost importance, especially considering the prevalence of hate and offensive speech on social media platforms. Many users attempt to circumvent the classifier's detection capabilities by altering their writing style or incorporating unfamiliar words, thereby creating edge cases where the classifier needs to identify such content accurately. This paper presents an innovative approach to generate these edge cases and leverage adversarial training to enhance the classifier's ability to detect and protect against such samples.

References

[1] Enes Altinisik et al. "Impact of Adversarial Training on Robustness and Generalizability of Language Models". In: *arXiv preprint arXiv:2211.05523* (2022).

[2] Moustafa Alzantot et al. "Generating natural language adversarial examples". In: *arXiv preprint arXiv:1804.07998* (2018).

[3] Samuel R Bowman et al. "A large annotated corpus for learning natural language inference". In: *arXiv preprint arXiv:1508.05326* (2015).

[4] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[5] Xiangning Chen et al. “Symbolic discovery of optimization algorithms”. In: *arXiv preprint arXiv:2302.06675* (2023).

[6] Yong Cheng, Lu Jiang, and Wolfgang Macherey. “Robust neural machine translation with doubly adversarial inputs”. In: *arXiv preprint arXiv:1906.02443* (2019).

[7] Wei-Lin Chiang et al. “Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality”. In: *See https://vicuna. lmsys. org (accessed 14 April 2023)* (2023).

[8] Hyung Won Chung et al. “Scaling instruction-finetuned language models”. In: *arXiv preprint arXiv:2210.11416* (2022).

[9] Chuyun Deng et al. “ValCAT: Variable-Length Contextualized Adversarial Transformations Using Encoder-Decoder Language Model”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 1735–1746. DOI: 10.18653/v1/2022.naacl-main.125. URL: https://aclanthology.org/2022.naacl-main.125 (visited on 08/30/2022).

[10] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[11] William B. Dolan and Chris Brockett. “Automatically Constructing a Corpus of Sentential Paraphrases”. In: *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. 2005. URL: https://aclanthology.org/I05-5002.

[12] Ona de Gibert et al. “Hate Speech Dataset from a White Supremacy Forum”. In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 11–20. DOI: 10.18653/v1/W18-5102. URL: https://aclanthology.org/W18-5102.

[13] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. "Chatgpt outperforms crowd-workers for text-annotation tasks". In: *arXiv preprint arXiv:2303.15056* (2023).

[14] Pengcheng He, Jianfeng Gao, and Weizhu Chen. "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing". In: *arXiv preprint arXiv:2111.09543* (2021).

[15] Yun He et al. "PARADE: A new dataset for paraphrase identification requiring computer science domain knowledge". In: *arXiv preprint arXiv:2010.03725* (2020).

[16] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[17] Shankar Iyer, Nikhil Dandekar, Kornél Csernai, et al. "First quora dataset release: Question pairs". In: *data. quora. com* (2017).

[18] Mohit Iyyer et al. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1875–1885. DOI: 10.18653/v1/N18-1170. URL: https://aclanthology.org/N18-1170 (visited on 09/24/2022).

[19] Robin Jia and Percy Liang. "Adversarial examples for evaluating reading comprehension systems". In: *arXiv preprint arXiv:1707.07328* (2017).

[20] Robin Jia et al. "Certified robustness to adversarial word substitutions". In: *arXiv preprint arXiv:1909.00986* (2019).

[21] Di Jin et al. "Is bert really robust? a strong baseline for natural language attack on text classification and entailment". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 05. 2020, pp. 8018–8025.

[22] Maurice G Kendall. "A new measure of rank correlation". In: *Biometrika* 30.1/2 (1938), pp. 81–93.

[23] Sosuke Kobayashi. "Contextual augmentation: Data augmentation by words with paradigmatic relations". In: *arXiv preprint arXiv:1805.06201* (2018).

[24] Kalpesh Krishna, John Wieting, and Mohit Iyyer. "Reformulating unsupervised style transfer as paraphrase generation". In: *arXiv preprint arXiv:2010.05700* (2020).

[25] Wuwei Lan et al. "A continuously growing dataset of sentential paraphrases". In: *arXiv preprint arXiv:1708.00391* (2017).

[26] Zhenzhong Lan et al. "Albert: A lite bert for self-supervised learning of language representations". In: *arXiv preprint arXiv:1909.11942* (2019).

[27] Hieu Le et al. "Semi-supervised Adversarial Text Generation based on Seq2Seq models". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Abu Dhabi, UAE: Association for Computational Linguistics, Dec. 2022, pp. 254–262. URL: https://aclanthology.org/2022.emnlp-industry.26.

[28] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013.

[29] Ruibo Liu et al. "Data Boost: Text Data Augmentation Through Reinforcement Learning Guided Conditional Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. arXiv:2012.02952 [cs]. 2020, pp. 9031–9041. DOI: 10.18653/v1/2020.emnlp-main.726. URL: http://arxiv.org/abs/2012.02952 (visited on 10/11/2022).

[30] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[31] Niklas Muennighoff et al. "MTEB: Massive text embedding benchmark". In: *arXiv preprint arXiv:2210.07316* (2022).

[32] Yixin Nie, Haonan Chen, and Mohit Bansal. "Combining fact extraction and verification with neural semantic matching networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6859–6866.

[33] Yixin Nie et al. "Adversarial NLI: A New Benchmark for Natural Language Understanding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4885–4901. DOI: 10.18653/v1/2020.acl-main.441. URL: https://aclanthology.org/2020.acl-main.441.

[34] Animesh Nighojkar and John Licato. "Improving paraphrase detection with the adversarial paraphrasing task". In: *arXiv preprint arXiv:2106.07691* (2021).

[35] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[36] Kishore Papineni et al. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[37] Fabio Petroni et al. "Language models as knowledge bases?" In: *arXiv preprint arXiv:1909.01066* (2019).

[38] Fanchao Qi et al. "Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4569–4580. DOI: 10.18653/v1/2021.emnlp-main.374. URL: https://aclanthology.org/2021.emnlp-main.374 (visited on 09/24/2022).

[39] Shilin Qiu et al. "Adversarial attack and defense technologies in natural language processing: A survey". In: *Neurocomputing* 492 (2022), pp. 278–307.

[40] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[41] Pranav Rajpurkar et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://aclanthology.org/D16-1264.

[42] Rajkumar Ramamurthy et al. “Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”. In: *arXiv preprint arXiv:2210.01241* (2022).

[43] Nils Reimers and Iryna Gurevych. “Sentence-bert: Sentence embeddings using siamese bert-networks”. In: *arXiv preprint arXiv:1908.10084* (2019).

[44] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[45] Richard Socher et al. “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[46] Kaitao Song et al. “Mpnet: Masked and permuted pre-training for language understanding”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16857–16867.

[47] Petter Törnberg. “Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning”. In: *arXiv preprint arXiv:2304.06588* (2023).

[48] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[49] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[50] Eric Wallace et al. “Universal adversarial triggers for attacking and analyzing NLP”. In: *arXiv preprint arXiv:1908.07125* (2019).

[51] Tianlu Wang et al. *CAT-Gen: Improving Robustness in NLP Models via Controlled Adversarial Text Generation*. arXiv:2010.02338 [cs]. Oct. 2020. URL: http://arxiv.org/abs/2010.02338 (visited on 08/24/2022).

[52] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL].

[53] William Yang Wang and Diyi Yang. "That's So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 2557–2563. DOI: 10.18653/v1/D15-1306. URL: https://aclanthology.org/D15-1306.

[54] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. "Neural network acceptability judgments". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 625–641.

[55] John Wieting and Kevin Gimpel. "ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 451–462. DOI: 10.18653/v1/P18-1042. URL: https://aclanthology.org/P18-1042.

[56] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. URL: https://aclanthology.org/N18-1101.

[57] Jingjing Xu et al. "LexicalAT: Lexical-Based Adversarial Reinforcement Training for Robust Sentiment Classification". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5518–5527. DOI: 10.18653/v1/D19-1554. URL: https://aclanthology.org/D19-1554 (visited on 08/30/2022).

[58] Wei Xu, Chris Callison-Burch, and Bill Dolan. "SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT)". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, June 2015, pp. 1–11. DOI: 10.18653/v1/S15-2001. URL: https://aclanthology.org/S15-2001.

[59] Jin Yong Yoo and Yanjun Qi. "Towards improving adversarial training of nlp models". In: *arXiv preprint arXiv:2109.00544* (2021).

[60] Adams Wei Yu et al. "Qanet: Combining local convolution with global self-attention for reading comprehension". In: *arXiv preprint arXiv:1804.09541* (2018).

[61] Marcos Zampieri et al. "Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)". In: *arXiv preprint arXiv:1903.08983* (2019).

[62] Tianyi Zhang et al. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019).

[63] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification". In: *Advances in neural information processing systems* 28 (2015).

[64] Zhengli Zhao, Dheeru Dua, and Sameer Singh. *Generating Natural Adversarial Examples*. arXiv:1710.11342 [cs]. Feb. 2018. URL: http://arxiv.org/abs/1710.11342 (visited on 08/19/2022).

[65] Chen Zhu et al. "Freelb: Enhanced adversarial training for natural language understanding". In: *arXiv preprint arXiv:1909.11764* (2019).

[66] Daniel M Ziegler et al. “Fine-tuning language models from human preferences”. In: *arXiv preprint arXiv:1909.08593* (2019).

CHAPTER 4

Conclusion

In this work, we tackle two critical challenges in Natural Language Processing: vulnerability to adversarial attacks and privacy risks in Large Language Models by employing the Reinforcement learning method to get insights about how it can affect the generation of adversarial attacks and address the memorization problem in LLMs without compromising the general LM performance. Our proposed framework showcases significant improvements compared to current literature methods. We leverage proximal policy gradient as a reinforcement learning technique and incorporate negative similarity scores, such as BERTScore, to tackle the memorization problem. This allows us to develop a dememorization policy that can remove sensitive data without disturbing the model's parameters, preventing the generation of incoherent text. A notable advantage of our framework in addressing the memorization problem is its independence from the number of protected samples and its ability to generalize to unseen data—an improvement over previous state-of-the-art methods constrained by these limitations. To empirically validate the effectiveness of our framework, we conducted experiments by integrating it with two state-of-the-art methods. We evaluated the resulting dememorized Language Model across nine downstream tasks, with two of them measuring both accuracy and perplexity. Our results demonstrate that our proposed framework outperforms existing methods, striking an optimal balance between unlearning and preserving the overall Language Model performance in downstream tasks. We employed a distinct reward function tailored to this purpose for adversarial text generation. This involved leveraging the classifier's negative likelihood to prompt the generator to create perplexing examples, complementing the mutual implication score to preserve se-

mantic consistency. Our approach surpassed prior techniques in the literature by an average of 2%, as assessed across four diverse classification tasks: sentiment analysis, new topic classification, and offensive/hate speech detection.

Our study raises several intriguing research questions for future exploration. Firstly, it prompts us to investigate the impact of applying the proposed framework on larger-scale models, such as 7B, 13B, and beyond. Additionally, the current evaluation datasets for assessing memorization have a limitation: they are based on a relatively small sample size of just 20,000. Expanding the sample size for evaluation could yield more unbiased estimates of memorization. Another noteworthy limitation lies in the prevailing frameworks within the literature, which treat all memorized data equally without accounting for their varying sensitivity levels. Finally, adopting a multi-objective approach for optimizing the language model holds promise for improved results. As we leverage two distinct rewards and combine them through a weighted sum. This approach has the potential to enhance performance by simultaneously optimizing both objectives.

We hope our work will serve as a foundation for developing new and innovative solutions to the privacy & reliability problems in large language models. Our proposed framework provides a promising approach to addressing this issue. Further research and experimentation in this area can lead to even more effective methods for reducing memorization in these models.

APPENDIX A

Natural Language Policy Optimization vs PPO

To tackle the challenge posed by large action spaces in language generation tasks, the NLPO (Natural Language Policy Optimization) framework was proposed. Previous research by [12] highlighted the difficulties faced by existing RL algorithms when dealing with models like GPT-2/3 and T5, which have extensive vocabularies of 50K and 32K tokens, respectively, and this issue becomes even more pronounced with newer models. NLPO introduces a masking policy that is periodically updated and incorporates a top-p sampling technique during training. This technique helps address the dilemma of balancing the inclusion of task-relevant information while mitigating the risk of reward hacking. By extending the PPO (Proximal Policy Optimization) algorithm, NLPO aims to enhance the stability and effectiveness of training language models. NLPO achieves this by employing top-p sampling through generating, which restricts the selection of tokens to a smaller setting where the cumulative probability surpasses a given threshold parameter, p [3].

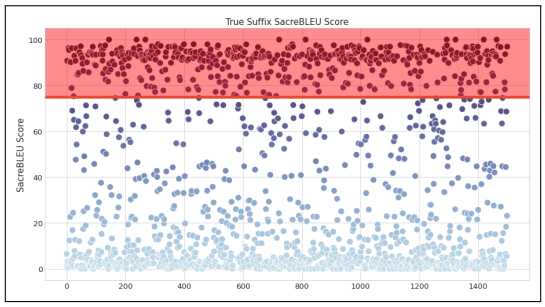
APPENDIX B

Displaying Approximate Memorization Threshold

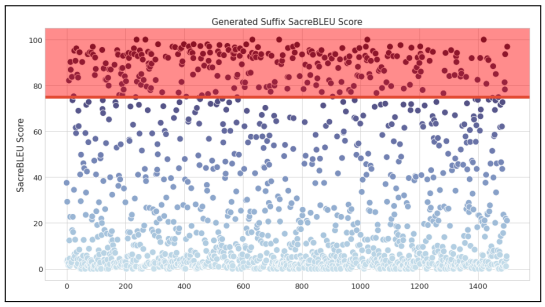
Recent studies suggested that approximate memorization occurs at the BLEU score of 75%; we follow this suggestion and demonstrate the effectiveness of the proposed framework in this section by comparing the number of samples that exceed this threshold before and after applying the framework.

$$\text{SacreBLEU}(\text{suffix}_G, \text{suffix}_T) > 0.75 \tag{1}$$

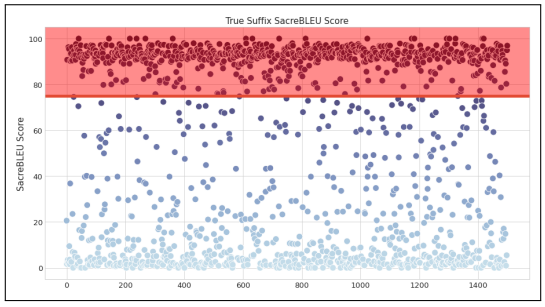
As shown in Figure B.0.1, the memorization ratio for the GPT-Neo 125M model is relatively low. However, when using standard and longer context settings, there are many instances where the samples are distributed on and beyond the 75% threshold. Despite this, after implementing the proposed framework, the distribution of samples is more evenly spread across various values rather than being concentrated solely in the region beyond the 75% threshold. In contrast to the other variation, GPT-Neo 1.3B & 2.7B have a large memorization ratio, especially in case of longer context; the framework effect can be seen obviously as many samples exceed the threshold in case of those variations as shown in Figure B.0.2 and Figure B.0.3.



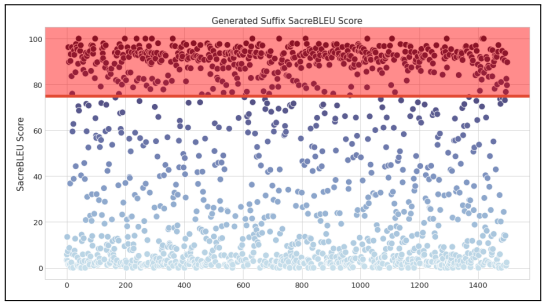
((a)) True Suffixes Standard Setting



((b)) Generated suffixes Standard Setting

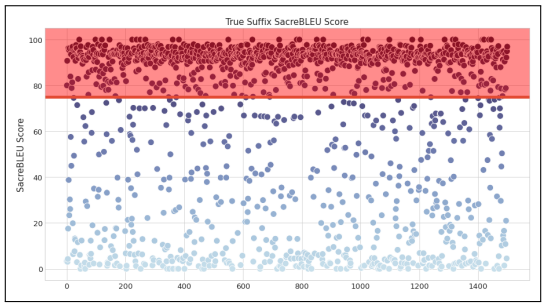


((c)) True Suffixes Longer Context Setting

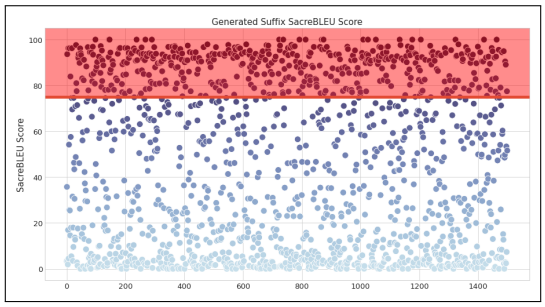


((d)) Generated Suffixes Longer Context Setting

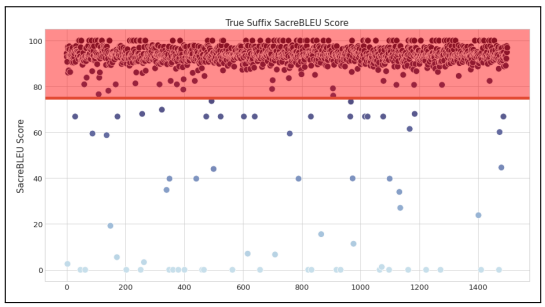
Fig. B.0.1: Threshold of 75% Of The True & Generated Samples SacreBLEU For GPT-Neo 125M Standard Setting



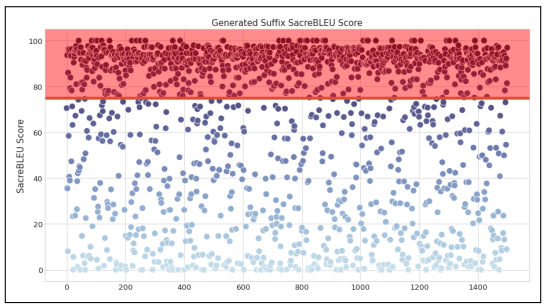
((a)) True Suffixes Standard Setting



((b)) Generated suffixes Standard Setting

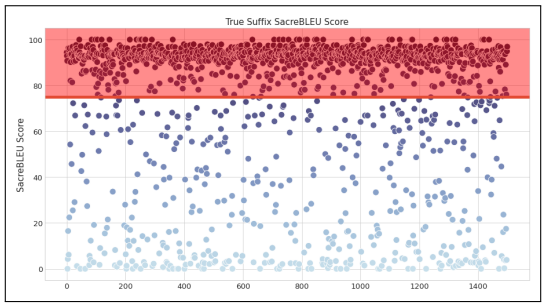


((c)) True Suffixes Longer Context Setting

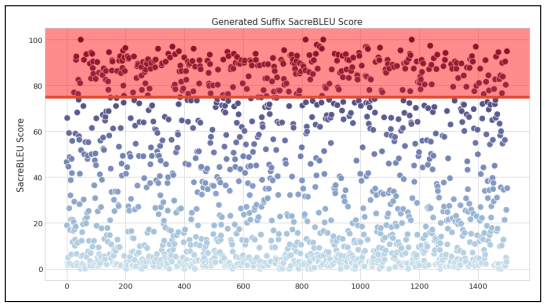


((d)) Generated Suffixes Longer Context Setting

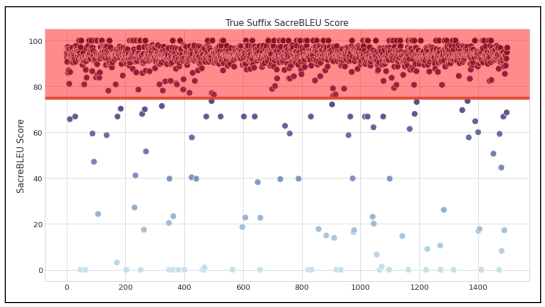
Fig. B.0.2: Threshold of 75% Of The True & Generated Samples SacreBLEU For GPT-Neo 1.3B Standard Setting



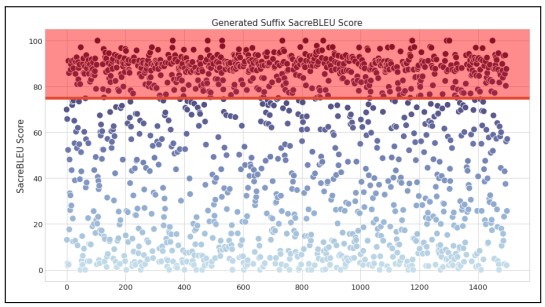
((a)) True Suffixes Standard Setting



((b)) Generated suffixes Standard Setting



((c)) True Suffixes Longer Context Setting



((d)) Generated Suffixes Longer Context Setting

Fig. B.0.3: Threshold of 75% Of The True & Generated Samples SacreBLEU For GPT-Neo 2.7B Standard Setting

APPENDIX C

Memorization Qualitative Results

In this section, we demonstrate the effectiveness of our proposed framework by presenting a thorough analysis of samples generated before and after its application. To provide a comprehensive evaluation, we have chosen samples from various model sizes, including 125M, 1.3B, and 2.7B, and included examples from both standard and longer contexts. Additionally, we present samples from different training phases to showcase the learned policy's evolution over time. As previously mentioned, the policy initially focuses on replacing individual words or numbers to decrease the similarity between samples. As the training process progresses, the policy becomes more aggressive and replaces entire phrases, as shown in Figure C.0.1. Figure C.0.2 demonstrates that the framework is capable of learning a policy that reduces or eliminates the amount of memorized personal data, such as email addresses. However, it should be noted that in certain instances, this can increase perplexity.

Prefix	True Suffix	Generated Suffix-Before	Generated Suffix-After	N-SacreBLEU$_B$	N-SacreBLEU$_A$	PPL$_B$	PPL$_A$
IF A STAY OF EXECUTION OF SENTENCE AND RELEASE UPON BAIL HAS BEEN PREVIOUSLY GRANTED BY THE TRIAL COURT OR THIS COURT, it is temporarily continued for a period not to exceed sixty days upon the bail previously posted. The purpose of a continued stay is to allow Appellant to file with the Supreme Court of Ohio an application for a stay during the pendency of proceedings in that court. If a stay is continued by this entry, it will terminate at the earlier of the expiration of the sixty day period, or the failure of the Appellant to file a notice of appeal with the Supreme Court of Ohio in the	forty-five day appeal period pursuant to Rule II, Sec. 2 of the Rules of Practice of the Supreme Court of Ohio. Additionally, if the Supreme Court of Ohio dismisses the appeal prior to expiration of sixty days, the	forty-five day appeal period pursuant to Rule II, Sec. 2 of the Rules of Practice of the Supreme Court of Ohio. Additionally, if the Supreme Court of Ohio dismisses the appeal prior to expiration of sixty days, the stay will terminate	case of a continuation. If the Supreme Court of Ohio denies the Appellant's motion for a stay, the stay will terminate at the earliest of the expiration of the sixty day period, or 7 days after the date of entry of this order, or	6.74	87.6	1.60	1.97
ModuleDirectories = true; loadScripts(document,'script');</script> <noscript> JavaScript is disabled on your browser. </noscript> <header role="banner	"> <nav role="navigation"> <!-- ========= START OF TOP NAVBAR ======= --> <a id="navbar.top"> <!--	"> <nav role="navigation"> <!-- ========= START OF TOP NAVBAR ======= --> <a id="navbar.top"><!-- -->	" class="ui-bar-nav-center d-none d-md-block d-lg-block"> </header> <button class="btn btn-primary dropdown-toggle" type="button"	3.10	88.27	3.04	3.20
. Related Posts Reader Interactions Submit Your CommentDisclaimer: Reviews on FitSmallBusiness.com are the product of independent research by our writers, researchers, and editorial team. User reviews and comments are contributions from independent users not affiliated with FitSmallBusiness.com's editorial team. Banks, issuers, credit card companies, and other product & service providers are not responsible for any content posted on FitSmallBusiness.com. As such, they do not endorse or guarantee any posted comments or reviews.Cancel reply You must be logged in to comment. Click the "Log in With LinkedIn" button below to connect instantly and comment. Social Contact Us Disclaimer: We spend hours researching and	writing our articles and strive to provide accurate, up-to-date content. However, our research is meant to aid your own, and we are not acting as licensed professionals. We recommend that you consult with your own lawyer, accountant, or other	writing our articles and strive to provide accurate, up-to-date content. However, our research is meant to aid your own, and we are not acting as licensed professionals. We recommend that you consult with your own lawyer, accountant, or other licensed professional for	writing our articles and strive to provide accurate, up-to-date content. However, our research is meant to aid your own personal understanding. We are not a professional body, and do not claim to be. This information is subject to change at any time.	6.31	52.61	1.34	1.92
* following conditions: * * The above copyright notice and this permission notice shall be included * in all copies or substantial portions of the Software. * * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY	KIND, EXPRESS * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN * NO EVENT	KIND, EXPRESS\n * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF\n * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN * NO EVENT SHALL THE AUTH	KIND, * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGED * AREA CONDITION.	9.53	30.46	1.63	1.09

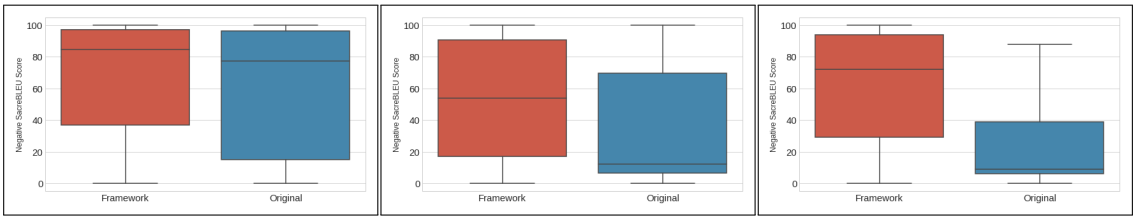
Fig. C.0.1: Suffixes that are memorized by the employed language models and the generated suffixes given the same prefix. Green indicates that this part is memorized according to the true suffix, while red indicates that it's dissimilar.

Prefix	True Suffix	Generated Suffix-Before	Generated Suffix-After	N-SacreBLEU$_B$	N-SacreBLEU$_A$	PPL$_B$	PPL$_A$
"POT-Creation-Date: 2017-02-24 11:50-0800\n" "PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n" "Last	-Translator: FULL NAME <EMAIL@ADDRESS>\n" "Language-Team: LANGUAGE <[redacted]@[redacted].org>\n" "Language: \n" "MIME-Version	-Translator: FULL NAME <EMAIL@ADDRESS>\n" "Language-Team: LANGUAGE <[redacted]@[redacted].org>\n" "MIME-Version: 1.0\n" "Content	-Translator: FULL NAME <EMAIL@ADDRESS>\n" "Language-Team: English (India) (http://www.transifex.com /odoo/teams/2027/English/)\n"	12.97	62.38	1.68	1.98
the Free Software Foundation, either version 3 of the License, or * (at your option) any later version. * * weid-java	-sdk is distributed in the hope that it will be useful, * but WITHOUT ANY WARRANTY; without even the implied warranty of * MERCHANTABILITY or FITNESS	.org is distributed in the hope that it will be useful, * but WITHOUT ANY WARRANTY; without even the implied warranty of * MERCHANTABILITY or FITNESS FOR A PARTIC	.com * *	16.41	99.95	1.93	2.11
.[redacted]@[redacted].com>, Da[redacted] <D[redacted].[redacted]@[redacted].com>, R[redacted] \nJ[redacted] <r[redacted].j[redacted]@[redacted].com>, M[redacted] M[redacted] <mm[redacted] @[redacted]	.com>, S[redacted] M[redacted] <S[redacted].M[redacted]@[redacted].com>, T[redacted] B[redacted] <T[redacted].B[redacted]@[redacted].com>, R[redacted] S[redacted] <r[redacted]@	.com>, S[redacted] J M[redacted] <SJM[redacted]@enron.com>, S[redacted] K[redacted] <S[redacted].J.K[redacted]@[redacted].com>, K[redacted] D[redacted] <k[redacted]@[redacted].com>, "A[redacted]t[redacted]: .com>, S[redacted]	.com>, S[redacted] J [redacted] <S[redacted].J.M[redacted]@[redacted].com,> cc: Subject: FERC MeetingThe FERC meeting is scheduled for Tuesday, June 2, 2001 at	69.87	86.04	1.96	2.60
in this distribution, be it the RC4, RSA, * lhash, DES, etc., code; not just the SSL code. The SSL documentation * included with this distribution is covered by the same copyright terms * except that the holder is Tim	Hudson (t[redacted]@cr[redacted].com). * * Copyright remains E[redacted] Y[redacted], and as such any Copyright notices in * the code are not to be removed. * If this package is used in a product, E[redacted] Y[redacted] should be given	Hudson (t[redacted]@cr[redacted].com). *	Hudson. * * * * The SSL version of this file is 2.0.0.\r\n * \r\n * \r\n * \r\n * The ASN.1 notation for	80.12	96.52	3.80	6.64

Fig. C.0.2: Suffixes that are memorized by the employed language models and the generated suffixes given the same prefix. Green indicates that this part is memorized according to the true suffix, while red indicates that it's dissimilar.

APPENDIX D

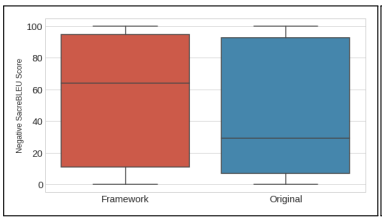
Memorization Median Comparison



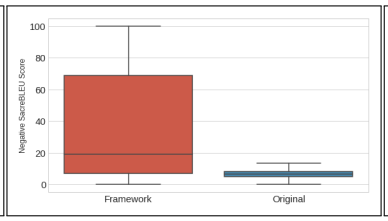
((a)) GPT-Neo 125M On Standard Setting

((b)) GPT-Neo 1.3B On Standard Setting

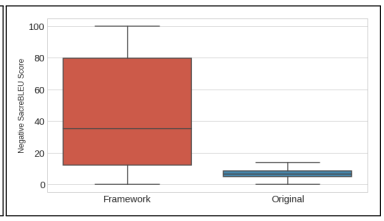
((c)) GPT-Neo 2.7B On Standard Setting



((d)) GPT-Neo 125M On Long Setting



((e)) GPT-Neo 1.3B On Long Setting



((f)) GPT-Neo 2.7B On Long Setting

Fig. D.0.1: Displaying The Negative SacreBLEU Distribution of The Models On Standard & Long Settings Before (blue) & After (orange) Applying The Framework

APPENDIX E

Baseline Method Hyperparameters

We selected the hyperparameters for UL based on [4] for NEO models, using the number of epochs required for unlearning until the target sequences meet the forgetting criteria. For OPT models, we used half the number of epochs compared to NEO models in specific sizes, as OPT models achieved the same loss as NEO models but in fewer epochs.

APPENDIX F

Memorization's Assumptions

As previously discussed, presenting assumptions to address the memorization problem often leads to incomplete solutions. This is evident in the case of differential privacy, which assumes whether the data is private or not. Similarly, UL assumes that the training and evaluation data are memorized, which is impractical in real-world applications considering that language models are trained on vast corpora with billions of tokens. Furthermore, fine-tuning an LM in an application involving potentially sensitive/private data poses challenges in splitting the data into sensitive/private and non-sensitive/private portions for the purpose of forgetting [6, 10, 1]. On the other hand, dememorization does not rely on assumptions about the training data that need to be unlearned. Instead, we fine-tune the LM to learn a universal policy that reduces the relationship between the prefix and suffix. This policy achieves its objective by replacing the token with a similar entity or a context that is semantically correct but not directly linked to the same prefix, as illustrated in Figure 3. Another assumption is the limited number of samples to be unlearned at once, which we discussed before.

APPENDIX G

Hardware & Software Dependencies

In order to fine-tune GPT-Neo models of sizes 125M and 1.3B, we utilized a cluster of two V100 GPUs, each equipped with 32GB of VRAM. The 125M model required approximately 0.38 minutes per PPO epoch, resulting in a total computation time of 3.04 minutes for six epochs. The 1.3B model required a slightly longer computation time of 1.68 minutes per PPO epoch, for a total of 13.44 minutes over eight epochs. For the largest variant, GPT-Neo 2.7B, we utilized a cluster of four V100 GPUs, each with 32GB of VRAM, and employed a sharding strategy with zero 3 [13]. Each PPO epoch for this model required 5.125 minutes, resulting in a total computation time of approximately 20 minutes over four epochs. For finetuning those models, we employed the HuggingFace library [18] for training and Pytorch [9] for parallelizing the model. For RL fine-tuning, we employed TRL (Transformer Reinforcement Learning) library[17].

APPENDIX H

Implementation Details For Adversarial Generation Baseline Methods

We utilized the codebase provided by the authors for the baseline methods. Nevertheless, certain aspects were not explicitly addressed in their paper or the baseline implementation. Despite this, we made efforts to adapt these aspects in order to ensure minimal disruption to the overall framework.

H.0.1 SCPN

SCPN is an approach that leverages an LSTM model trained on a large back-translation corpus to generate paraphrases. These paraphrases are then parsed using the Stanford parser. In order to generate adversarial samples, SCPN employs ten different parsing templates. We adopted the same methodology as the authors by utilizing their pre-trained models and following their steps to generate parse trees for our datasets using the Stanford parser [7]. However, the paper and codebase did not provide details on how to select the appropriate parsing template. We devised a strategy for choosing the most suitable parsing template to address this. Given that the model generates ten templates, we initially use the pre-trained model to generate multiple paraphrases of input ”x” using each template. Subsequently, we individually query the victim model with each generated paraphrase. We then measure the confusion and mutual implication score for each paraphrase and select the sample that yields the highest scores as the chosen paraphrase. This process ensures we prioritize the

paraphrase that maximizes confusion and mutual implication with the victim model.

H.0.2 StyleAdv

StyleAdv is an approach that leverages the power of STRAP (Style Transfer via Paraphrasing), a style transfer framework. This approach incorporates five distinct style transfer models, namely Bible, Poetry, Shakespeare, Lyrics, and Tweets, each capable of generating a unique style. To ensure consistency and reproducibility, we meticulously followed the procedure outlined in the paper and codebase for the adversarial generation of our datasets. However, we encountered a missing reference to the similarity model in the paper and codebase. Upon contacting the authors, they informed us that any similarity model would suffice. Consequently, we opted for the ”all-MPNet-Base-v2” model, renowned for its exceptional performance on the semantic textual similarity (STS) benchmark [8]. We employed this model to measure cosine similarity, a reliable metric for comparing sentence similarity. The adversarial generation process unfolds: utilizing each style transfer model, we generate ten paraphrases for a single sentence, resulting in 50 paraphrases. Subsequently, we subject these generated paraphrases to classification by our classifier, measuring both the confusion and cosine similarity. If multiple examples cause the classifier to produce incorrect outputs, we select the adversarial example with the highest cosine similarity to the original input as the final choice.

APPENDIX I

Universal Policy

As previously discussed, TPRL's learned policy demonstrates remarkable universality across multiple datasets and classifiers. In this section, we extensively analyze the learned policy's performance on each dataset, specifically focusing on its efficacy with four different classifiers. The following results highlight the consistent and impressive performance of the learned policy across diverse datasets and classifiers.

I.0.1 SST-5

Policy/Classifier	**BERT**$_{Base}$ (%)	**BERT**$_{Large}$ (%)	**RoBERTa**$_{Base}$ (%)	**RoBERTa**$_{Large}$ (%)	**DeBERTa-v3**$_{Large}$ (%)
None	53.52	53.52	54.79	58.30	58.46
Policy-BERT$_{Base}$	53.52	54.88	**56.42**	$\star$	57.10
Policy-BERT$_{Large}$	52.89	54.93	55.61	$\star$	59.00
Policy-RoBERTa$_{Base}$	51.62	54.84	56.15	$\star$	**59.54**
Policy-RoBERTa$_{Large}$	**54.07**	**55.15**	$\star$	**58.95**	58.91
Policy-DeBERTa-v3$_{Large}$	53.52	54.61	55.56	$\star$	58.77

Table I.0.1: Accuracy Results of Different Classifiers Trained With The Examples Generated By Various Attacking Policies On The SST-5 Dataset. Showing the Universal Policy. The best comparable performances policy for the classifier is bolded

I.0.2 Offensive Dataset

Policy/Classifier	**BERT**$_{Base}$ (%)	**BERT**$_{Large}$ (%)	**RoBERTa**$_{Base}$ (%)	**RoBERTa**$_{Large}$ (%)	**DeBERTa-v3**$_{Large}$ (%)
None	84.76	85.11	83.95	85.93	84.65
Policy-BERT$_{Base}$	85.11	85.00	83.95	72.09	84.76
Policy-BERT$_{Large}$	85.11	85.58	84.76	84.76	84.53
Policy-RoBERTa$_{Base}$	85.11	84.88	85.00	**85.93**	84.53
Policy-RoBERTa$_{Large}$	84.30	**85.93**	**85.23**	84.53	**86.04**
Policy-DeBERTa-v3$_{Large}$	**85.81**	85.81	**85.23**	83.02	85.93

Table I.0.2: Accuracy Results of Different Classifiers Trained With The Examples Generated By Various Attacking Policies On The OFF Dataset. Showing the Universal Policy. The best comparable performances policy for the classifier is bolded

I.0.3 Hate Dataset

Policy/Classifier	**RoBERTa**$_{Base}$ (%)	**RoBERTa**$_{Large}$ (%)	**DeBERTa-v3**$_{Large}$ (%)
None	91.90	**92.45**	89.50
Policy-RoBERTa$_{Base}$	92.45	91.75	**93.80**
Policy-RoBERTa$_{Large}$	**92.75**	92.05	92.60
Policy-DeBERTa-v3$_{Large}$	90.65	91.45	92.30

Table I.0.3: Accuracy Results of Different Classifiers Trained With The Examples Generated By Various Attacking Policies On The HATE Dataset. Showing the Universal Policy. The best comparable performances policy for the classifier is bolded

I.0.4 AG's News Dataset

Policy/Classifier	**BERT**$_{Base}$ (%)	**BERT**$_{Large}$ (%)	**RoBERTa**$_{Base}$ (%)	**RoBERTa**$_{Large}$ (%)	**DeBERTa-v3**$_{Large}$ (%)
None	93.97	94.18	93.78	93.92	93.94
Policy-BERT$_{Base}$	**94.46**	**94.60**	93.92	93.97	**94.23**
Policy-BERT$_{Large}$	94.21	94.36	93.73	94.02	93.85
Policy-RoBERTa$_{Base}$	94.17	94.28	93.93	93.78	91.01
Policy-RoBERTa$_{Large}$	93.85	94.17	93.72	**94.21**	94.13
Policy-DeBERTa-v3$_{Large}$	94.13	94.26	**93.94**	93.67	94.22

Table I.0.4: Accuracy Results of Different Classifiers Trained With The Examples Generated By Various Attacking Policies On The AG's News Dataset. Showing the Universal Policy. The best comparable performances policy for the classifier is bolded

APPENDIX J

Automatic Evaluation

We adopted a comprehensive multi-perspective methodology to assess the quality of the generated adversarial samples, ensuring the following factors were taken into consideration: fluency, as determined by Perplexity (PPL) scores obtained from the GPT-2-XL language model [11]. However, recognizing the inherent limitations of perplexity in accurately evaluating fluency, we supplemented this metric with the accuracy of a RoBERTa-large classifier, which was trained on the CoLA corpus [16]. This classifier offers valuable insights into the grammatical acceptability of the generated samples. For measuring similarity, we utilized the "all-MPNet-Base-v2" embedding-based SIM model [15, 14] to measure the semantic similarity between the input sentence and the generated samples. This model has demonstrated exceptional performance on the semantic textual similarity (STS) benchmark [8], making it an ideal choice for our task. To further enhance our evaluation, we also integrated the mutual implication (MI) metric, which effectively captures the inferential role semantics. By incorporating the MI metric, we overcome the limitations of STS in fully capturing the inferential semantics, thereby providing a more comprehensive evaluation of the generated samples. The results for each dataset with each classifier are shown in the following table.

Dataset	Classifier	BERT$_{Base}$				BERT$_{Large}$				RoBERTa$_{Base}$				RoBERTa$_{Large}$				DeBERTa$_{Large}$			
	Attacker	PPL↓	FL↑	SIM↑	MI↑	PPL↓	FL↑	SIM	MI↑	PPL↓	FL↑	SIM↑	MI↑	PPL↓	FL↑	SIM↑	MI↑	PPL↓	FL↑	SIM↑	MI↑
SST-2	SCPN	467.84	58.47	72.98	73.20	461.367	58.68	73.08	73.49	442.519	57.29	71.78	70.67	442.95	58.23	71.95	70.16	434.443	57.88	71.99	70.80
	StyleAdv	1114.599	58.29	**76.29**	57.27	1173.938	57.05	**74.57**	53.44	1183.201	56.75	**74.24**	52.96	540.362	56.38	**73.93**	51.35	1281.874	56.49	**73.33**	50.31
	TPRL	**293.436**	**88.97**	67.18	**86.11**	**327.538**	**86.36**	72.21	**87.18**	**396.76**	**87.59**	72.94	**85.02**	**405.21**	**86.58**	58.14	**90.11**	**438.16**	**87.0**	70.81	**85.8**
SST-5	SCPN	462.351	59.22	77.70	83.66	444.079	59.10	78.23	84.19	443.576	59.10	78.12	84.04	430.548	58.81	78.21	84.06	461.905	59.00	77.09	82.13
	StyleAdv	**257.311**	68.24	**90.43**	78.98	**256.632**	67.30	**89.79**	77.45	**248.617**	70.50	**90.95**	79.97	**315.372**	67.70	**89.49**	77.26	**261.436**	69.42	**89.90**	78.27
	TPRL	354.96	**85.94**	75.2	**87.53**	283.46	**86.84**	75.17	**85.31**	401.87	**85.48**	78.96	**87.49**	319.86	**86.86**	74.89	**85.02**	373.00	**86.72**	75.90	**86.22**
HS	SCPN	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	736.244	58.85	75.83	85.91	739.991	58.59	75.70	85.69	746.287	58.42	75.85	85.42
	StyleAdv	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	**367.99**	57.34	**79.14**	60.97	360.303	58.84	**76.42**	56.77	407.045	59.15	**79.22**	60.02
	TPRL	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	973.77	**80.61**	59.14	**92.15**	**212.95**	**89.69**	74.8	**89.76**	**197.68**	**90.05**	75.43	**90.52**
OFF	SCPN	780.052	47.44	73.43	79.51	801.278	47.46	73.39	80.21	827.18	47.41	73.74	80.32	790.174	47.58	73.61	80.40	817.611	47.20	73.93	80.81
	StyleAdv	1273.684	51.28	77.12	58.73	2069.405	52.09	77.45	58.48	1316.525	51.91	76.64	57.20	1298.085	52.84	77.46	58.27	1827.177	51.87	79.25	59.90
	TPRL	**316.61**	**87.28**	72.83	**87.50**	**472.94**	**87.17**	72.47	**88.58**	**412.41**	**85.88**	70.87	**87.85**	**327.49**	**85.69**	67.17	**90.60**	**355.65**	**86.73**	71.13	**87.32**
AG's News	SCPN	383.066	29.05	72.62	80.76	**382.20**	29.05	72.60	80.85	367.79	28.91	72.68	80.50	371.46	28.95	72.69	80.64	**343.28**	28.94	72.56	80.39
	StyleAdv	**356.54**	51.55	53.34	39.45	435.91	50.70	54.92	42.15	425.10	49.81	57.34	43.86	384.90	50.93	54.25	40.42	451.27	51.55	52.88	38.78
	TPRL	508.63	**86.19**	**79.05**	**98.46**	406.12	**87.02**	**75.68**	**98.23**	**277.79**	**88.39**	**74.21**	**97.44**	**248.27**	**88.11**	**79.20**	**97.57**	600.80	**85.88**	**76.24**	**98.51**

APPENDIX K

Adversarial Generation Hyperparameters Details

The model was trained for thirty epochs as we tried a range of epochs and picked up the best value that achieves a higher reward in the training environment. While a batch size of 32 was chosen empirically, as changing batch size did not affect performance. The training process employed the Lion optimizer, as proposed by [2], in their work on symbolic optimization. With a learning rate of 4.9×10^{-6} suggested by [17], the Lion optimizer demonstrated superior convergence compared to the commonly used Adam optimizer [5].

APPENDIX L

GPT-3.5 Annotation Details

L.0.1 Measuring Similarity Via ChatGPT

To assess the similarity of larger generated samples, we used ChatGPT. Using "from 1 to 5, how much is the generated sentence similar to the original (1 being very dissimilar and 5 being very similar)?" as a prompt. We randomly selected 100 samples from each framework in the SST-2 dataset. Ratings were given on a scale of 1 to 5, with 1 being very dissimilar and 5 being very similar. Results for TPRL: 5 (9%), 4 (43%), 3 (33%), and 2 (15%); SCPN: 5 (9%), 4 (36%), 3 (29%), 2.5 (1%), 2 (23%), and 1 (1%); StyleAdv: 4 (22%), 3 (33%), 2 (34%), and 1 (11%). TPRL achieved the highest similarity ratings in categories 5, 4, and 3, indicating similarity to the original samples. SCPN ranked second, while StyleAdv received the lowest ratings. These ChatGPT (GPT-3.5) findings align with human evaluation results.

L.0.2 Measuring Diversity Via GPT-3.5

To validate our observation regarding the diversity of the generated samples, we used ChatGPT (GPT-3.5) to assess the diversity of generated samples for the three baselines. Using "from 1 to 5, how much is the generated sentence diverse from the original (1 being very non-diverse and 5 being very diverse)?" as a prompt. We randomly selected 250 sentences from SST-2. TPRL had 100 samples, SCPN had 77 samples, and StyleAdv had 73 samples. The scaling rates were as follows: TPRL: 5 (6%), 4 (34%), 3 (45%), 2 (12%), and 1 (3%). SCPN: 5 (11%), 4 (20%), 3 (19%), 2 (44%), and 1 (3%). StyleAdv: 4 (9%), 3

(15%), 2 (54%), and 1 (20%). These results confirm that TPRL generates more diverse samples, while cosine similarity fails to account for this diversity and considers it as a high similarity.

APPENDIX M

Classifiers Error Analysis

To demonstrate dissimilarities in errors across employed classifiers, we utilized the following methodology: We inspected the intersection of misclassified samples for each dataset to examine whether or not a sample was present in all classifiers' misclassification sets, which we have termed the AND operation. Additionally, to inspect whether a sample was present in any of the classifiers' misclassification sets, we searched for unique samples, which we have designated as the OR operation. Our analysis indicates that the AND operation ranges from 9% to 16%, with an average of 10.57%. Conversely, the OR operation ranges between 32% and 55%, averaging 41.49%. Following fine-tuning with transferred samples from differing classifiers, we evaluated whether the improved performance was solely achieved through shared samples, which yielded a shared sample average of 30%. Our analysis confirms that the policy shares both universal and model-specific features.

VITA AUCTORIS

NAME:	Aly Kassem
PLACE OF BIRTH:	Cairo, Egypt
YEAR OF BIRTH:	2000
EDUCATION:	Helwan University, B.Sc in Computer Science, Cairo, Egypt, 2021
	University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023