Authorship Anonymization: Differentially-private Text Generation and Writing Style Transfer

Haohan Bo

Master of Science

School of Computer Science

McGill University

Montreal, Quebec

2019-10-01

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Science

ACKNOWLEDGMENTS

First and foremost, I would like to gratefully acknowledge my supervisor, Benjamin C. M. Fung. Professor Fung is an excellent and admirable supervisor who has provided me with patient guidance, support, and the freedom to pursue my own research ideas. In addition he has given me sound advice and support on my career. I often consider myself lucky to be supervised by Professor Fung and to be a member in the Data Mining and Security (DMaS) Lab.

I also would like to thank Dr. Steven H. H. Ding - my colleague and friend. He spent endless time in discussions with me and advising on the research. Our weekly meeting has been the highlight of my week. He has not only provided me with the support on research but also has shared his experiences and life with me. He has cheered me up whenever I got frustrated with the research progress. It is my greatest pleasure to work with you. May your family find great happiness in Kingston.

I am also grateful to all my friends I met in Canada. We have had a great and enjoyable time in this fascinating place. I would like to thank Yu Qie and Justine Salam for helping me with the French abstract.

Last but not least, I would like to express my gratitude to my family. Thank you for your unconditional support, understanding, love, and company.

ABSTRACT

Privacy is a major concern for information gathering and data publishing across a wide range of online applications. Various privacy-preserving algorithms and models have been extensively studied for relational data, network graph data, trajectory data, and transactional data, etc. Among them, text data is the most prevalent unstructured data on the Internet, but the studies on sanitizing textual data are still preliminary. Most privacy protection studies for textual data focus on removing explicit sensitive identifiers. However, personal writing style, a strong indicator of authorship, is often neglected. Most works focus only on removing or replacing explicit sensitive phrases or personal identifiers in the text. Text data, such as anonymous peer review comments or product reviews, carries an implicit personal trait: writing style. Modern stylometric techniques can identify the actual author of a given anonymous text snippet from 10,000 candidates. However, only a few works are proposed for writing style anonymization, and the ones that satisfy privacy requirements only treat text as numeric vectors that are difficult for the recipients to interpret.

To tackle this problem we propose two novel text generation models for authorship anonymization. Combined with a semantic embedding reward loss function and the exponential mechanism, our proposed auto-encoder can generate differentially-private texts that have a close semantic and similar grammatical structure to the original text while removing personal traits of the writing style. It does not require any conditioned labels or paralleled text data during training. Another model uses

a generative adversarial network with back-translation loss function; the model is able to hide the authorship by imitating the writing style of a reference dataset. We evaluate the performance of the proposed models on the real-life peer reviews dataset and the Yelp review dataset. The result suggests that our models outperform the state-of-the-art on semantic preservation, authorship obfuscation, and stylometric transformation.

ABRÉGÉ

La confidentialit est une proccupation cardinal pour lensemble dinformations et la publication de donnes sur une grande gamme dapplications en ligne. Divers algorithmes et modles prservant la confidentialit ont t largement tudis pour les donnes relationnelles, les donnes de graphe de rseau, les donnes de trajectoire, les donnes transactionnelles, etc. Parmi eux, les donnes textuelles sont les donnes non structures les plus rpandues sur Internet, mais les tudes sur la dsinfection des donnes textuelles sont encore prliminaires. La plupart des tudes de protection de la confidentialit des donnes textuelles se concentrent sur la suppression des identifiants sensibles et explicites. Cependant, le style d'criture personnel, comme un indicateur fort d'auteur, est souvent nglig. La majorit des travaux se concentrent uniquement sur la suppression ou le remplacement de phrases sensibles ou d'identifiants personnels explicites dans le texte. Les donnes textuelles, telles que les commentaires anonymes par les pairs ou les critiques de produits, comportent un trait personnel implicite: le style d'criture. Les techniques stylomtriques modernes permettent d'identifier l'auteur rel d'un extrait de texte anonyme parmi 10 000 candidats. Pourtant, seules quelques uvres sont proposes pour l'anonymisation du style d'criture, et celles qui rpondent aux exigences de confidentialit ne traitent le texte que comme des vecteurs numriques difficiles interprter par les destinataires.

Pour rsoudre ce problme, nous proposons deux nouveaux modles de gnration de texte pour l'anonymisation de la paternit. Combin une fonction de perte de rcompense incorpore smantique et au mcanisme exponentiel, notre encodeur automatique propos

peut gnrer des phrases diffrentiellement-prives qui possdent une structure smantique et grammaticale proche du texte original, tout en supprimant les traits personnels du style dcriture. Il nexige pas dtiquettes conditionnes ni de donnes textuelles en parallle pendant la formation. Un autre modle utilise un rseau contradictoire gnratif avec une fonction de perte de traduction arrire; le modle peut masquer la paternit en imitant le style d'criture d'un jeu de donnes de rfrence. Nous valuons la performance des modles proposs sur l'ensemble de donnes de la critique relle et sur celle de Yelp. Le rsultat suggre que nos modles surpassent ltat de la technique en matire de conservation smantique, dobscurcissement de lauteur et de transformation stylomtrique.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Privacy is a vital issue in online data gathering and public data release. Various machine learning models and privacy preservation algorithms have been studied for relational data [Johnson et al., 2018], network graph data [Chen et al., 2014], trajectory data [Chow and Mokbel, 2011], and transactional data [Li et al., 2012]. Some of them have been successfully adopted in real-life applications such as telemetry collection [Cortés et al., 2016]. However, the studies on privacy protection for textual data are still preliminary. Most related works focus only on replacing the sensitive key phrases in the text [Anandan et al., 2012, Sanchez et al., 2013, Vasudevan and John, 2014] without considering the author's writing style, which is a strong indicator of an author's identity. Even though some textual data, such as double-blind academic reviews, is released anonymously, adversaries may recover an author's identity using the personal traits in writing. Stylometric techniques [Koppel et al., 2011] can identify an author of a text from 10,000 candidates. The techniques are effective across online posts, articles, emails, and reviews [Zheng et al., 2006, Ding et al., 2015, 2017]. Nevertheless, traditional text sanitization methods [Narayanan and Shmatikov, 2008] focus on anonymizing the contents, such as patient information, instead of the writing style, so they are ineffective against writing style analysis.

The original author can be easily re-identified even if protected by these traditional approaches [Narayanan and Shmatikov, 2008].

Only a few recent studies focus on authorship anonymization, aiming to hide the personal traits of writing style in the given textual data. *Anonymouth* [McDonald et al., 2012] is a semi-automatic framework that offers suggestions to users to change their writing style. Yet, this framework is not practical because it requires two datasets as a reference to compare the change in writing style. Also, the user has to make all the final modification decisions. *SynTF* [Weggenmann and Kerschbaum, 2018] represents a line of research that protects the privacy of the numeric vector representation of textual data. It adopts the exponential mechanism for privacy guarantee, but the output is only an opaque term frequency vector, not an interpretable text in natural language. Furthermore, its token substitution approach does not consider the grammatical correctness and semantic aspects.

Style transfer is another line of research that tries to generate text with controllable attributes [Shen et al., 2017, Hu et al., 2017, Sennrich et al., 2016, Yang et al., 2018, Chen et al., 2018]. Representative models [Hu et al., 2017, Logeswaran et al., 2018] can control the sentiment and tense of the generated text. However, they do not modify the personal traits in writing. Their applications on sentiment and word-reordering correspond to the content of the text more than the writing style. We argue that their definition of styles, such as sentiment or tense, is different from the personal linguistic writing characteristics that raise privacy concerns. *A4NT* [Shetty et al., 2018] is a generative neural network that sanitizes the writing style of the input text. However, it requires text samples to be labeled with known

author identities. It is not applicable to any textual data. Additionally, according to the samples provided in the paper, it has difficulties keeping the same semantic meaning between the original and the generated text.

To address the aforementioned issues, we propose two models, one of which is an *Embedding Reward Auto-Encoder (ER-AE)* to generate differentially-private text. It protects an author's identity through text distinguishability. ER-AE does not assume any labels nor any parallel data. Relying on differential privacy, its privacy protection is independent of an adversary's background knowledge and does not assume any specific adversarial scenarios. ER-AE receives the original text as input, extracts latent features, and generates a new text using the exponential mechanism. Inspired by the *REINFORCE* algorithm [Sutton et al., 2000], we include a semantic embedding reward loss function. It is able to keep the generated text a close semantic similarity to the original while protecting the text privacy.

Another model is a *Generative Model with Back-Translation (GM-BT)*, which hides the authorship of given text through transferring the input text into the writing style of the reference dataset. After processing, machine learning authorship identifiers are not able to distinguish the real authorship. In our experiments, a subset of Wikipedia dataset is utilized to be the reference dataset in the experiment because it has a mixture of writing style and written by different authors. GM-BT does not require any label or paired data in the training set. All the input tokens are first encoded into a feature vector, the same as ER-AE. Then, the feature vector is concatenated with the reference writing style embedding as a latent vector. Based on the information in the latent vector, the generator produces new text that has

a writing style similar to the reference dataset. We adopt the adversarial learning method and back-translation loss function to train the generator. The generator can anonymize the input by changing the text writing style while preserving the semantic similarity.

1.1 Contributions

Unlike the aforementioned authorship anonymization works, both ER-AE and GM-BT produce human-friendly text in natural language. Our key contributions are summarized as follows:

Contributions of ER-AE Model:

- This is the first differentially-private authorship anonymization model that is able to generate human-friendly text in natural language instead of a numeric vector.
- We present a sequential text generator that employs exponential mechanism to protect privacy through a sampling process.
- We propose a new semantic reward function that is able to better preserve the semantic and sentiment similarity between the original and the generated text.
- We provide a theoretical analysis on the privacy properties of the proposed model. By analyzing our proposed model ER-AE under the recently proposed concept of document indistinguishability [Fernandes et al., 2018], we observe that a large privacy budget is necessary to balance the utility and author identity indistinguishability under the strictest adjacency definition. This observation also holds for other related work that builds on a similar concept of document indistinguishaiblity.

- Comprehensive evaluations on two real-life datasets, namely *NeurIPS & ICLR peer reviews* and *Yelp product reviews*, suggest that ER-AE is effective in both anonymizing the writing styles and preserving the semantics of the original text.

Contributions of GM-BT Model:

- We propose a generative model without sampling operation to anonymize text through writing style imitation.
- We combine the GAN framework with back-translation loss function to train the model in an unsupervised way without requiring any labels in the training dataset.
- Comprehensive evaluations on the same two datasets suggest that GM-BT is also effective in both anonymizing the text and preserving the semantics of the original text. In addition, GM-BT, which does not have a sampling operation, can produce more human-friendly texts than ER-AE.

1.2 Contribution of Authors

I am responsible for all the implementations and experiments in all chapters. I discussed the problem definitions and the methodologies with Professor Benjamin C. M. Fung and Professor Steven H. H. Ding.

1.3 Thesis Organization

This thesis is organized as follows: Chapter 2 presents recent works related to our research topic and technologies. Chapter 3 proposes a differentially-private text generation model to protect an individual's privacy and presents the details

and results of conducted experiments. Chapter 4 elaborates our second model that protects privacy through writing style transfer via adversarial learning method and back-translation loss function. Chapter 5 concludes this thesis and discusses the future works.

CHAPTER 2
Related Work

2.1 Differential Privacy

Differential privacy has recently received a lot of attention in the privacy protection and machine learning communities. The differentially-private deep learning model [Abadi et al., 2016b] and the deep private auto-encoder [Phan et al., 2016] are designed to preserve the training data privacy. Their purpose is to guarantee that publishing the trained model does not reveal the privacy of individual records. Our purpose is different. In our first model, we publish the differentially-private data generated by the model, rather than the model itself. Most existing models for differentially-private data release, such as Chen et al. [2014], Dankar and El Emam [2012], focus on different types of data rather than text. One recent work [Weggenmann and Kerschbaum, 2018] aims to protect privacy in text data using the exponential mechanism. However, it releases the term frequency vectors instead of a readable text. This approach limits the utility of published data to only the applications that assume term frequency as features. In contrast, the goal of our proposed ER-AE model is to generate differentially-private text in a natural language while anonymizing the writing styles.

2.2 Writing Style Features

Writing style features usually contain a set of linguistic marks that represent the writing characteristics of an author that could be used to distinguish different authors. Various types of features have been introduced [Stamatatos, 2009], such as lexical-level features, character-level features, syntactic features, etc. Many of them are context-free manually-crafted stylometric features, for example, a set of features including text length, word length, and word frequency are included in the study of [Rudman, 1997]; also, the frequency of special tokens (e.g., /, @), the number of upper case, and the position of quoted content are adopted in the framework proposed by [Zheng et al., 2006]. In addition to manually-crafted styles, a representation learning algorithm to learn stylometric through deep learning model in an unsupervised way is proposed by [Ding et al., 2017]. In this thesis, by using the features proposed in works [Iqbal et al., 2013] and [Zheng et al., 2006], we utilize the distance of the stylometric features vectors to measure the change of writing style.

2.3 Generative Adversarial Networks

The *Generative Adversarial Networks (GANs)* [Goodfellow et al., 2014] framework is a hot topic in recent research on generative machine learning. It is a framework to learn a generative model through an adversarial learning method without paralleled training data. The work of conditional generative adversarial networks [Mirza and Osindero, 2014] is able to control the content of the generated image based on the given label through an extra discriminator to guide the generator. A text to image generative model called *StackGAN* [Zhang et al., 2017] is

proposed to translate given text to image by utilizing the matching-aware discriminator. Their purpose is to generate a specific type of image data that is continuous, according the given conditional information. However, our goal is to generate discrete textual data.

2.4 Controllable Text Generation

Text generation is a trending topic in machine learning. It aims at generating a text sample with changed attributes. Sennrich et al. [2016] propose a model to change the degree of politeness while generating text. Hu et al. [2017] combine the *variational auto-encoders (VAE)* with *generative adversarial network (GAN)* to generate a text with different sentiment and tense. *A4NT* [Shetty et al., 2018] is able to control the gender and age attribute of the generated text data through a GAN model. Most of the literature on this direction name different attributes, such as sentiment and tense, as style. However, these attributes correspond more to the content itself, rather than the personal writing style. Our focus is different; we pay more attention to the change of linguistic writing style.

2.5 Image Style Transfer

The style transfer on an image has been widely discovered by recent works [Gatys et al., 2016, Wang and Gupta, 2016, Zhu et al., 2017, Isola et al., 2017]. Generally, there are two ways to transfer the style of image. Gatys et al. [2016] propose a framework to directly extract content and style information in the image and generate the image with a specific style based on this information, whereas Wang and Gupta [2016], Zhu et al. [2017], Isola et al. [2017] combine GAN with a convolutional neural

network to transfer image to image via designing an adversary to guide the generator. The key challenge in those projects is that it is non-trivial to verify the quality of the content and style of the generated image. Due to the discreteness of textual data, the style and content are even more difficult to determine in our task.

2.6 Writing Style Transfer

A substantial amount of research focuses on style transfer for text data. They can be broadly categorized as writing style imitation, semi-automatic anonymization, and the author's attribute transfer. Although there are some methods that work on sentiment transfer and claim themselves as "style transfer", from the perspective of authorship analysis, we do not consider sentiment as a writing style feature based on the description of Stamatatos [2009]. Studies on writing style transferal try to change the writing style revealed from the text according to a given author. Shetty et al. [2018] design a GAN to transfer Obama's text to Trump's style. A sequence-to-sequence (seq2seq) model is proposed by Jhamtani et al. [2017] to transfer modern English into Shakespearean English. Shen et al. [2017] design a model with a cross-alignment method to control the sentence sentiment while preserving semantic. These models can also be applied in writing style anonymization. However, these studies require the data to be labeled with authorship identity. They assume a number of known authors. In contrast, our differentially-private and imitation solutions do not assume any label information.

2.7 Writing Style Obfuscation

Writing style obfuscation studies try to hide the identity of the author. Anonymouth [McDonald et al., 2012] is a tool that utilizes *JStylo* to generate writing attributes. It gives users suggestions on how they can anonymize their text according to two reference datasets. Kacmarcik and Gamon [2006] also propose a similar architecture to anonymize text. Instead of directly changing the text, they all work on the term frequency vector, whose real-life utility is limited. Compared to semi-automatic methods that require users to make a decision, our approach directly learns from the raw textual data, providing an end-to-end solution.

CHAPTER 3
Differentially-private Text Generation

In this chapter, we present our first model, Embedding Reward Auto-Encoder (ER-AE), to anonymize text. To protect the authorship of a given text, ER-AE aims to generate differentially-private text. The generated text is expected not to change the semantics of the given text. ER-AE consists of an auto-encoder with the exponential mechanism. We propose a novel loss function, embedding reward loss function, to help the model generate human-friendly text. A set of experiments are conducted on Yelp reviews dataset and Conferences' reviews dataset to evaluate the performance of ER-AE on aspects of the privacy protection and utility preservation. In addition, the intermediate results of the generator are shown to prove the impact of the embedding reward loss function.

3.1 Preliminaries

3.1.1 Auto-encoders

Auto-encoders are prevalent in recent works on conditional generation. Traditionally, an auto-encoder is designed to extract features from a large dataset automatically. Usually, it is trained through an unsupervised learning method from the

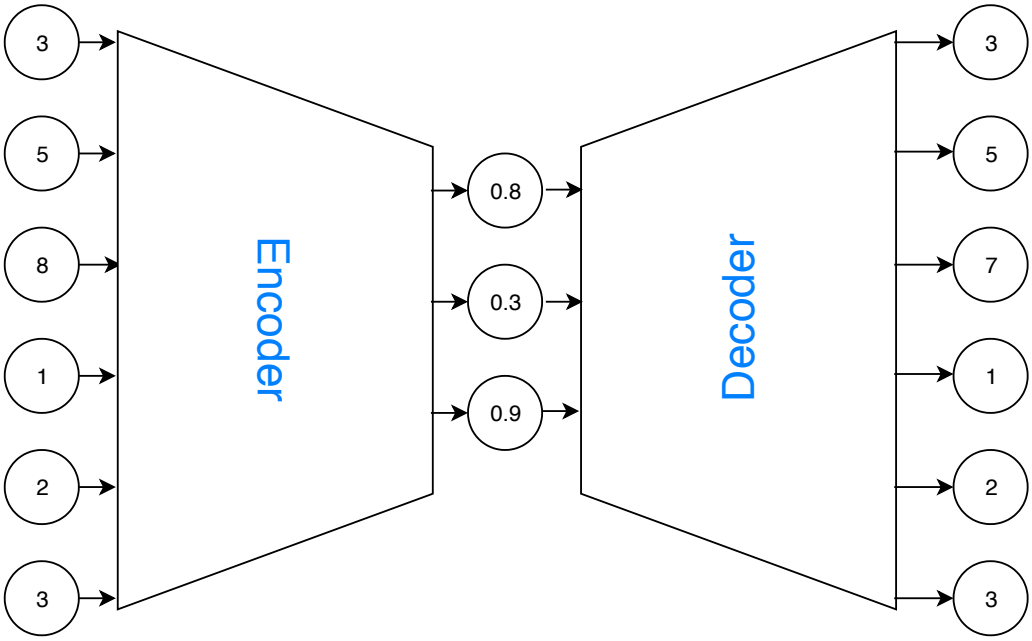

Figure 3–1: Overall architecture of auto-encoders.

training dataset without a label or paired data, which means it is learned automatically. An auto-encoder consists of an encoder and a decoder. As shown in Figure 3–1, by receiving the input data the encoder outputs a latent vector that represents the feature vector of input data, and the decoder reconstructs the original input data based on the feature vector. If the size is equal to or larger than the input data, the encoder can choose to cheat by producing exactly the same data, and it could only copy and not extract any information. Therefore, the size of the feature vector is usually designed to be smaller than the input data. In this case, the encoder is forced to learn to compress the received information and removes less important features. The encoder can be considered as a feature extractor.

Formally, let X stand for the input data. Define $E(.)$ as the encoder that extracts the input X into a feature vector, and $D(.)$ as a decoder that tries to reconstruct the input data based on the feature vector produced by the encoder. Originally, the decoder is to help the encoder produce the best feature vector for the input that the decoder can reconstruct the data. Then, we have the objective function while training a auto-enocoder through maximizing maximum likelihood:

$$\mathcal{L} = -\log D\left(X|E\left(X\right)\right). \tag{3.1}$$

By having the ability to compress the data, auto-encoders can also be used to remove noise from the data. In addition, the ability of the decoder to reconstruct the original input data based on the feature vector has many applications in data generation tasks.

3.1.2 Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of artificial neural networks that play an important role in processing temporal sequence data. RNNs can be applied to sequential data classification [Lai et al., 2015], sequential data generation [Yang et al., 2018], speech recognition [Graves et al., 2013], etc.

Traditionally, artificial neural networks consist of an input layer, one or more hidden layers, and an output layer. The hidden layer is usually a fully connected layer that connects every node in the previous layer to every node in the current layer. However, this type of hidden layer fails to capture the temporal information in the data because temporal information is lost while treating every node without

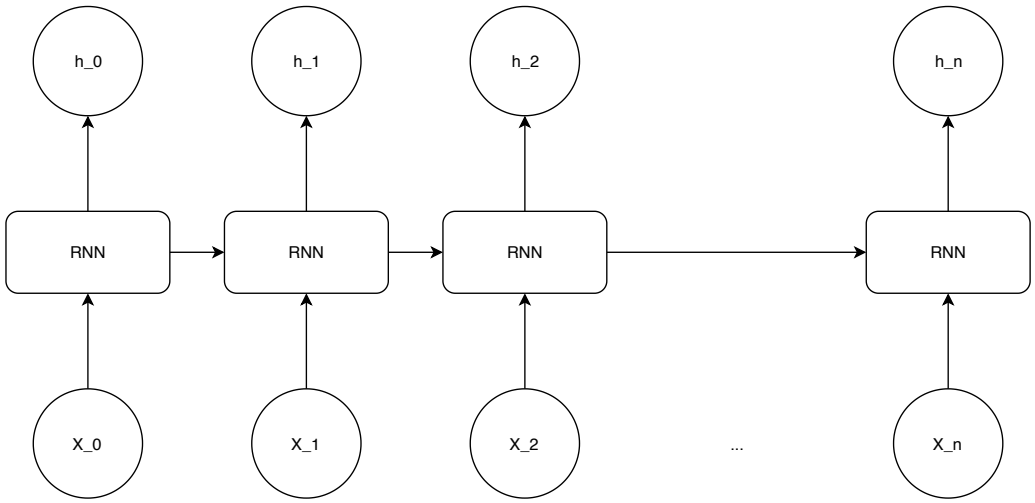

Figure 3–2: Overall architecture of RNN.

any temporal signal. RNN, as shown in Figure 3–2, maintains the sequential information in the input data through the recursive process. The temporal information is memorized in the RNN hidden state. Every time step, RNN receives the input data and updates its internal hidden state to store all received information. Then, it outputs the hidden state for further usage. For example, we can use the last hidden state to do classification or prediction via a fully connected layer, also called the many-to-one model.

Mathematically, given a sequential data X, X_t that stands for the t-th item in X, and the t-th initial hidden state s_t, we have the update function in t-th time step for the hidden state:

$$s_t = \phi(UX_t + Ws_{t-1}), \tag{3.2}$$

where ϕ is the activation function and U and W are the transformation matrices for x_t and s_{t-1}, respectively.

However, the training of traditional RNN is often non-trivial. There exists the vanishing or exploding gradient problem caused by the recursive feature of RNN while calculating the gradient based on the chain rule. According to Equation 3.2, we have $\frac{\delta s_t}{\delta W}$:

$$\begin{aligned}\frac{\delta s_t}{\delta W} =& \frac{\delta s_t}{\delta s_{t-1}} \frac{\delta s_{t-1}}{\delta s_{t-2}} \frac{\delta s_{t-2}}{\delta s_{t-3}} \cdots \frac{\delta s_1}{\delta s_0} \frac{\delta s_0}{\delta W} \\ =& \prod_{i=1}^{t-2} W \phi^{'}(U x s_i + W s_{i-1}) \frac{\delta s_0}{\delta W},\end{aligned}$$

where $\phi^{'}$ is the derivative of ϕ. Due to the large number of multiplication operations, we can see that if $\phi^{'}$ is smaller than 1, then the result of the whole equation tends to be zero, which is called the vanishing gradient. On the other hand, while $\phi^{'}$ is larger than 1 it becomes the gradient explosion problem. In addition, because the hidden state is updated in every time step by the same transformation matrices, the long-term information is hard to propagate, which causes the calculation of the output to be biased towards more recent inputs [Chung et al., 2015]. To overcome those issues, *Long Short-Term Memory (LSTM)* [Hochreiter and Schmidhuber, 1997] and *Gated Recurrent Unit (GRU)* [Cho et al., 2014b] are proposed. Their general ideas provide an alternative channel to propagate long-term yet important memory to the current state.

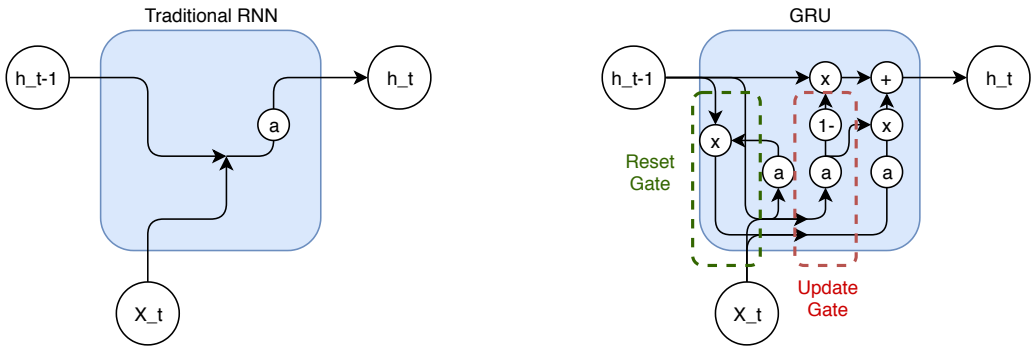

Figure 3–3: Overall architectures of a traditional RNN cell and GRU cell on how to produce a next hidden state. In the small circles, a is an activation function, x is a multiplication operation, $1-$ means one minus the input, and $+$ is an addition operation. The reset gate of GRU is in the region of the left dashed rectangle, and the update gate is in the region of the right dashed rectangle.

3.1.3 Gated Recurrent Unit

Gated Recurrent Unit (GRU) [Cho et al., 2014b] is a new type of recurrent neural network cell designed to solve the vanishing gradient and long-term memory problems that come with the traditional RNN mentioned in Section 3.1.2. Unlike the traditional RNN cell that only contains a hidden state, GRU (Figure 3–3) additionally adopts an update gate and a reset gate that help the network remember long-term information and avoid gradient vanishing and explosion. Generally, the two gates are represented as two vectors that decide what information should be forgotten and updated to produce the output.

To better explain the mechanism of GRU, we show the internal architectures of a traditional RNN cell and GRU cell in Figure 3–3. The traditional RNN is very simple and only has one operation to update the hidden state. However, the GRU is more complicated with reset gates and update gates to have better control of

the received information. The left dashed rectangle marks the reset gate; it decides what information in the past should be forgotten. At the same time, the update gate controls how much previous information should be kept, which is similar to the memory cell in LSTM. This design helps the model achieve a balance between long-term and short-term memory.

Formally, at time step t, given the previous hidden unit h_{t-1} and the input data X_t, the reset gate is defined as:

$$r_t = \sigma(W_r X_t + U_r h_{t-1}),$$

where σ is the logistic sigmoid function, and W_r and U_r are the transformation matrices of the reset gate. Recall that the reset gate decides how much information should be forgotten. We can get the temporal new hidden state:

$$\widetilde{h_t} = \phi(W X_t + U(r_t \odot h_{t-1})),$$

where ϕ is the tanh activation function. In this function, if the model decides to store all the previous information, then r_t is close to 1. On the other hand, when r_t is close to 0, the previous information is dropped, and the hidden state is "reset" by the recently received data.

Additionally, the updated date z_t is also calculated by the previous hidden state h_{t-1} and the input X_t:

$$z_t = \sigma(W_z X_t + U_z h_{t-1}).$$

After getting the z_t and $\widetilde{h}_t$, the new hidden state can be calculated as:

$$h_t = z_t h_{t-1} + (1 - z_t)\widetilde{h}_t. \tag{3.3}$$

As shown in Equation 3.3, the update gate z_t controls how much information from the previous state is kept in the current hidden state. More previous information is passed to the current state when z_t is larger, which assists the model to remember long-term information.

Utilizing the reset gate and update gate described above, GRU is able to capture long-term and short-term dependencies. While GRU prefers to extract short-term, the reset gate would mostly tend to be close to 1. On the other hand, while the long-term dependent information is needed, the update gate would tend to be close to 1 to pass more information from the previous hidden state to the next one.

Besides solving the long-term memory issue, the gradient vanishing and explosion problem can be avoided during the training of GRU [Fu et al., 2016]. Compared with LSTM, which can also solve those problems, GRU requires fewer computation resources and achieves a competitive performance [Cho et al., 2014a].

3.1.4 Policy Gradient Methods and REINFORCE

Policy gradient [Sutton et al., 2000] methods are a family of reinforcement learning methods that have been widely used in recent works. Silver et al. [2016] use Policy Gradient methods to train a network that can beat the best human Go player in the world. Yu et al. [2017] utilize Policy Gradient to solve the discrete data generation problem in GAN. Usually, a reinforcement learning method makes a decision based on the estimated value function, whereas the basic idea of Policy Gradient

is to directly learn a parameterized policy to make the decision without knowing a value function. Treating any differentiable function as a policy function, it can be optimized through back-propagating the gradient while maximizing the final reward.

Given a differentiable function as policy π with parameters θ and a preferred score function $J(\theta)$:

$$J(\theta) = v_{\pi_\theta}(s_0),$$

where s_0 is the initial state and v_{π_θ} is the value function under π_θ.

According to the result of Sutton and Barto [2018], we can find the gradient to optimize the policy:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta),$$

where the μ is the distribution of a state, q_π is the q value of a pair of state and action under policy π. One popular policy-gradient method is *REINFORCE* [Williams, 1992], which is the policy gradient with Monte-Carlo sampling. The training of a policy gradient requires simulated samples to calculate the gradient and optimize the policy. In REINFORCE it adopts the complete samples that have complete return

through *Monte-Carlo* sampling, then the gradient becomes:

$$\begin{aligned}
\nabla J(\theta) \propto & \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \\
= & \mathbb{E}_\pi \left[\sum_a q_\pi \left(S_t, a\right) \nabla \pi \left(a|S_t\theta\right) \right] \\
= & \mathbb{E}_\pi \left[\sum_a \pi \left(a|S_t,\theta\right) q_\pi \left(S_t, a\right) \frac{\nabla \pi \left(a|S_t\theta\right)}{\pi \left(a|S_t,\theta\right)} \right] \\
= & \mathbb{E}_\pi \left[q_\pi \left(S_t, A_t\right) \frac{\nabla \pi \left(A_t|S_t,\theta\right)}{\pi \left(A_t|S_t\theta\right)} \right] \\
= & \mathbb{E}_\pi \left[G_t \frac{\nabla \pi \left(A_t|S_t,\theta\right)}{\pi \left(A_t|S_t,\theta\right)} \right],
\end{aligned}$$

where, at t-th time step, G_t is the sample return, A_t is the action in the sample, and S_t is the state corresponding to the sample. Here, $J(\theta)$ becomes:

$$J(\theta) = \mathbb{E}_\pi[G_t \ln \pi \left(A_t|S_t,\theta\right)]. \tag{3.4}$$

Having the Equation 3.4, we can train a differentiable policy function end-to-end with Monte-Carlo sampling.

3.1.5 Differential Privacy

Differential privacy [Dwork et al., 2006] is a framework that provides a rigorous privacy guarantee on a dataset. To protect the privacy of an individual, it requires that the output from any analysis should not be sensitive to the change of a single record in a dataset.

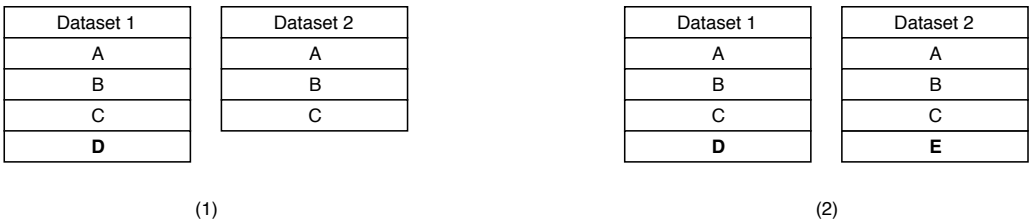

Figure 3–4: Examples of two definitions of adjacent datasets.

Compared with *k-anonymity* [Sweeney, 2002] and *ℓ-diversity* [Machanavajjhala et al., 2006], which assume hackers do not have access to extra identical features, differential privacy does not have any assumption on the attackers' background knowledge. Differential privacy has been applied in the data collection process by enterprises to obscure an individual's identity. For example, Apple uses a differential privacy mechanism to preserve users' privacy while collecting data from the iPhone so that they are able to extract patterns from a large group of users without compromising individual privacy [Cortés et al., 2016].

Adjacency is a key notion in differential privacy. In the original definition, as shown in the Figure 3–4(1), datasets D_1 and D_2 are considered to be adjacent if D_2 can be obtained by adding or removing one record in D_1. Another commonly used adjacency definition (Figure 3–4(2)) is that D_1 and D_2 are adjacent if D_2 can be obtained by modifying one record in D_1 [Dwork et al., 2010, Ding et al., 2018, Soria-Comas and Domingo-Ferrert, 2013, Soria-Comas et al., 2017].

Differential privacy requires *inherent randomness* of a sanitization algorithm or generation function:

Definition 3.1.1. Differential Privacy. Let $\epsilon > 0$ be a privacy budget. Let $\mathcal{A} : D^n \to Z$ be a randomized algorithm. An algorithm $\mathcal{A}$ preserves ϵ-differential

privacy if for any two adjacent datasets D_1, $D_2 \in D^n$, and for any possible set of output $Z \in im(\mathcal{A})$ where im produces the image of the input:

$$Pr\left[\mathcal{A}\left(D_1\right) \in Z\right] \leq e^{\epsilon} \cdot Pr\left[\mathcal{A}\left(D_2\right) \in Z\right] \qquad \square$$

Differential privacy guarantees that the result from a given algorithm $\mathcal{A}$ is not sensitive to a modification of any individual record in D, which makes adjacent datasets probabilistically indistinguishable. ϵ denotes the privacy budget, the allowed degree of sensitivity. Compared to k-anonymity [Sweeney, 2002] and ℓ-diversity [Machanavajjhala et al., 2006], differential privacy does not assume any attacker's background knowledge. A large ϵ implies a higher risk to privacy. However, ϵ is a relative value that implies different degrees of risk given different problems [Weggenmann and Kerschbaum, 2018]. Some studies use a large ϵ [Sala et al., 2011], while others use a smaller value [Chen et al., 2014].

3.2 Problem Definition

Adversary Scenario. Generally in an authorship identification problem, one assumes that the attacker holds an anonymous text authored by one of the suspects from the dataset. The attacker aims to infer the true author of the anonymous text based on a set of reference texts from each suspect.

However, this scenario assumes certain information on the applicable dataset, such as author labels and labeled reference text samples. To make our algorithm applicable to most datasets, we do not assume any labels in the dataset.

This leads to the the strictest and most conservative definition of adjacency. Following Weggenmann and Kerschbaum [2018], we define that any two texts can be considered as adjacent.

Adjacency for Text Data. Any two texts can be considered *adjacent* in the strictest scenario that datasets D_1 and D_2 both have only one record (text), and D_2 can be obtained by editing the only one record in D_1 following the second commonly used definition of adjacency in Section 3.1.5.

Unlike the original differential privacy setting that one wants to privately release the result of an algorithm on all or a subset of these records (n records input $\rightarrow$ 1 result output), our text anonymization problem is to release each anonymized text independently (1 text input $\rightarrow$ 1 text output), which leads to the strictest scenario that one dataset in the definition of adjacency only contains one text record. This one-record-per-dataset setting also corresponds to the framework of local differential privacy [Kasiviswanathan et al., 2011] that provides privacy guarantee between arbitrary two individual records rather than any two adjacent datasets with multiple records.

With differential privacy, we can have text indistinguishability: given adjacent texts, one can hardly distinguish the identity of any text to another. In our case, the identity of a text corresponds to the author that writes the text. Along with this, as shown in Figure 3–5, the attacker would fail in the original authorship identification scenario since the anonymous text is indistinguishable from the rest of texts.

There are works, such as Fernandes et al. [2018], that factor in a text's topic in the privacy model. However, the approach limits to the datasets that contain

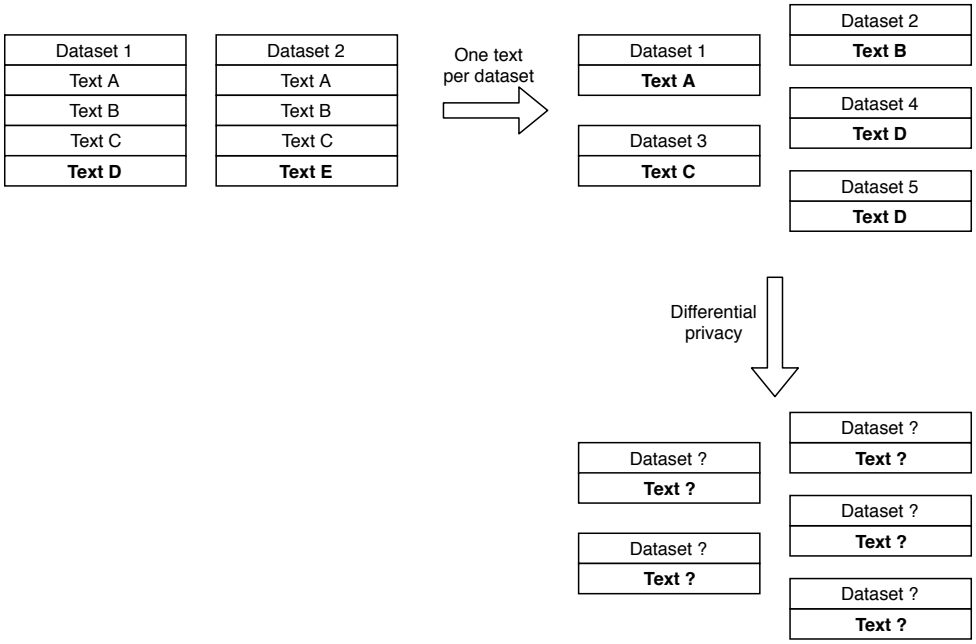

Figure 3–5: Differential privacy protects dataset identity.

explicit authorship and topic labels or indicators. Our definition follows Weggenmann and Kerschbaum [2018] leading to the strictest and most conservative definition of adjacency, which can apply to any text dataset.

With the concept of differential privacy we further define our problem:

Definition 3.2.1. Differentially-private Text Generation. Let $\mathbb{D}$ denote a dataset that contains a set of texts where $x \in \mathbb{D}$ is one of them and $|x|$, the length of the text, is bound by l. Given $\mathbb{D}$ with a privacy budget ϵ, for each x the model generates another text $\tilde{x}_{dp}$ that satisfies ϵl-differential privacy. □

Following the above definitions, any two datasets that contain only one record are probabilistically indistinguishable w.r.t. a privacy budget ϵ. It directly protects the identity of an individual record, disregarding if some of the records belong to the

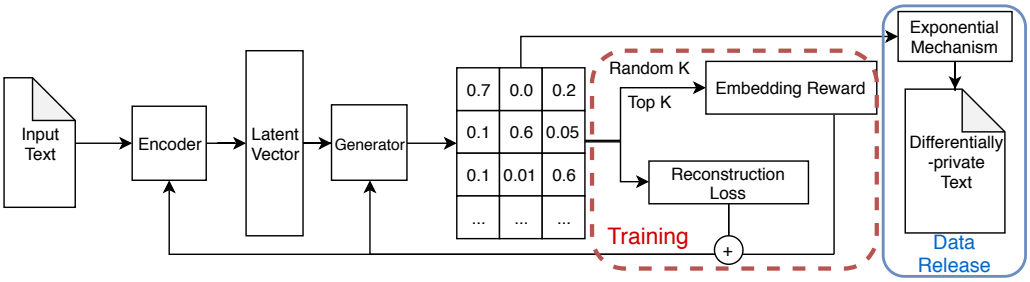

Figure 3–6: Overall architecture of ER-AE.

same author or not. It assumes that every record is authored by a different author, which is the strictest situation. Suppose that an author may write k different text records, by achieving text indistinguishability, one cannot tell if any two of these k text records are from the same author or not because they are all indistinguishable. In this way, my definition does not assume k to be any number. Thus, it is in the strictest form that can apply to any text dataset.

Technically, the proposed text generation approach protects the writing style by reorganizing the text, replacing tokens with different spelling, removing the lexical, syntactical and idiosyncratic features of the given text. The above problem definition is based on SynTF Weggenmann and Kerschbaum [2018], but our target is text in natural language rather than numeric vectors, which is more challenging.

3.3 ER-AE for Differentially-private Text Generation

In this section, we present our ER-AE model (see Figure 3–6). ER-AE contains an encoder and a generator. Its encoder receives a sequence of tokens as input and generates a latent vector to represent the semantic features. The generator,

combined with an exponential mechanism, is able to produce differentially-private text according to the latent vector. ER-AE is trained by combining a reconstruction loss function and a novel embedding loss function.

3.3.1 Bi-directional RNN Encoder

Our ER-AE model starts with a basic sequence-to-sequence (seq2seq) auto-encoder structure. Given a text x, its tokens $\langle x_1, \ldots, x_l \rangle$ are first converted into a sequence of embedding vectors $\langle Em(x_1), \ldots, Em(x_l) \rangle$ by $Em : \mathcal{V} \to \mathbb{R}^{m_1}$, where V is the vocabulary across the dataset, and m_1 is the embedding dimension. On the top of the embedding vectors, we apply a bi-directional recurrent neural network with *Gated Recurrent Unit (GRU)* [Cho et al., 2014a] that leverages both the forward and backward information. GRU achieves a comparable performance to LSTM but less computational overhead [Cho et al., 2014a]. Then, the produced final state vectors from both directions, $\boldsymbol{s}_f$ and $\boldsymbol{s}_b$, are concatenated and linearly transformed to be a latent vector $E(x)$. m is the hidden state dimension for the GRU function.

$$E(x) = \boldsymbol{W}_h \times \text{concat}(\boldsymbol{s}_f, \boldsymbol{s}_b), \text{ where } \boldsymbol{s}_f, \boldsymbol{s}_b \in \mathbb{R}^m, \ \boldsymbol{W}_h \in \mathbb{R}^{h \times 2m} \tag{3.5}$$

3.4 Text Generator

The generator is another recurrent neural network with GRU. It generates a text token-by-token. For each timestamp i, it calculates a logit weight z_{iv} for every candidate token $v \in \mathcal{V}$, conditioned on the latent vector, last original token x_{i-1}, and the last hidden state $\boldsymbol{s}_{i-1}$ of the GRU function.

$$z_{iv} = \boldsymbol{w}_v^\top \text{GRU}(E(x), Em(x_{i-1}), \boldsymbol{s}_{i-1}) + b_v \tag{3.6}$$

Let $\tilde{x}_i$ denote the random variable for the generated token at timestamp i. Its probability mass function is proportional to each candidate token's weight z_{ti}. This is modeled through a typical softmax function:

$$Pr[\tilde{x}_i = v] = \exp(z_{iv}) / \sum_{v' \in \mathcal{V}} \exp(z_{iv'}) \tag{3.7}$$

A typical seq2seq model generates text by applying $\text{argmax}_{v \in \mathcal{V}} Pr[\tilde{x}_i = v]$ for each timestamp i. However, this process does not protect the privacy of the original data.

3.4.1 Differentially-private Text Sampling

To protect an individual's privacy and hide the authorship of the original input text, we couple the *exponential mechanism* [McSherry and Talwar, 2007] with the above sampling process in the generator. The exponential mechanism can be applied to both numeric and categorical data [Fernandes et al., 2018]. It has been shown to be effective in various sampling processes for discrete data. It guarantees privacy protection by injecting noise into the sampling process:

Definition 3.4.1. Exponential Mechanism. Let $\mathcal{M}$ and $\mathcal{N}$ be two enumerable sets. Given a privacy budget $\epsilon > 0$, a rating function ρ: $\mathcal{M} \times \mathcal{N} \to \mathbb{R}$. The probability density function of the random variable $\varepsilon_{\epsilon,\rho}(m)$ is described as:

$$Pr\left[\varepsilon_{\epsilon,\rho}(m) = n\right] = \frac{\exp\left(\frac{\epsilon}{2\Delta}\rho(m,n)\right)}{\sum_{n'} \exp\left(\frac{\epsilon}{2\Delta}\rho\left(m,n'\right)\right)} \tag{3.8}$$

where Δ, the sensitivity, means the maximum difference of rating function values between two adjacent datasets, and $m \in \mathcal{M}$, $n \in \mathcal{N}$. □

The exponential mechanism protects privacy through disturbing the distribution of the rating function using ϵ, and the data is randomly sampled based on the disturbed distribution. This sampling process is ϵ-differentially private.

Following Weggenmann and Kerschbaum and by swapping $\mathcal{M}$ and $\mathcal{N}$ with text dataset $\mathbb{D}$ and our vocabulary $\mathcal{V}$. Assuming arbitrary $x \in \mathbb{D}, w \in \mathcal{V}$, and D_x stands for the dataset that only contains the text x, our rating function and its sensitivity for arbitary timestamp i can be described as:

$$\begin{aligned} \rho_i(D_x, w) &= \frac{\exp\left(z_{iw}\right)}{\sum_{w' \in \mathcal{V}} \exp\left(z_{iw'}\right)} \in [0, 1] \\ \Delta_{\rho_i} &= \max_{t \in \mathcal{V}} \max_{D_x \sim D_{x'}} ||\rho_i(D_x, t) - \rho_i(D_{x'}, t)||_1 \leq 1 \end{aligned} \tag{3.9}$$

The rating function seeks alternative tokens w to the original tokens in the input text dataset by considering the logit weight values from Equation. 3.6. It rates w by considering the current context and encoded latent features. By adopting this rating function, the sampling process considers both the grammatical and semantic context to find an alternative token that can preserve one's privacy. It is timestamp-specific, and its sensitivity is bounded by 1. Let $\varepsilon_{\epsilon,\rho_i}(\tilde{x}_i)$ denote the random variable for the generated token at timestamp i. By plugging our rating function into the exponential mechanism defined in Equation 3.8, we have the probability mass function for $\varepsilon_{\epsilon,\rho_i}(\tilde{x}_i)$:

$$Pr[\varepsilon_{\epsilon,\rho_i}(\tilde{x}_i) = v] = \frac{\exp\left(\frac{\epsilon}{2\Delta}\rho_i(D_x, v)\right)}{\sum_{v'} \exp\left(\frac{\epsilon}{2\Delta}\rho\left(D_x, v'\right)\right)} \tag{3.10}$$

This function models the disturbed probability distribution for all the alternative token v to replace the original one. According to Definition 3.4.1, sampling from

$\varepsilon_{\epsilon,\rho_i}(\tilde{x}_i)$ for each timestamp i is ϵ-differentially private. Recall that in Definition 3.1.1, the timestamp is bound by l. To generate text $\tilde{x}_{dp}$, the generator samples a token for each timestamp i through Equation 3.10:

$$\tilde{x}_{dp}[i] \sim \varepsilon_{\epsilon,\rho_i}(\tilde{x}_i) \quad \text{for } i \in [1, l] \tag{3.11}$$

The *composition theorem* [Dwork et al., 2014] (Theorem 3.16) is an extension to differential privacy. By repeating n ϵ-differentially-private algorithms, the complete process achieves an ϵn-differential privacy.

Theorem 1. ***Deferentially-Private Text Sampling*** *Given a privacy budget $\epsilon > 0$, and a sequence length $l > 0$, the generator's sampling function in Equation 3.11 is ϵl-differentially-private.*

Proof. At the generation stage, for each timestamp i, our model generates a token by sampling from Equation 3.10, which follows the form of the exponential mechanism. This process achieves ϵ-differential privacy as in Definition 3.4.1. By repeating this process l times, the complete sampling function provides ϵl-differential privacy. $\tilde{x}_{dp}$ is ϵl-differentially private. □

With Theorem 1, we can protect the text privacy by achieving text indistinguishability (Figure 3–5) and writing style protection as explained in Section 3.2.

3.4.2 Reconstruction Loss

In order to generate a human-friendly text that has a close semantic to the original text, we need to have a high-quality rating function ρ_i for Equation 3.9. This is achieved by training the ER-AE model's encoder to extract semantic information and

its generator to learn the relationships among the tokens for prediction. We follow an unsupervised learning approach since we do not assume any label information. First, we adopt the reconstruction loss function:

$$\mathcal{L}_{recon} = \sum_{x_i \in x, x \in \mathbb{D}} -\log Pr\left[\tilde{x}_i = x_i\right] \tag{3.12}$$

It maximizes the probability of observing the original token x_i itself for the random variable $\tilde{x}_i$. In the recent controllable text generation models, the reconstruction loss function plays an important role to preserve grammar structure and semantics of input data [Shetty et al., 2018, Shen et al., 2017] when combined with the other loss functions.

3.4.3 Training with Embedding Reward

Diving into the optimization aspect of the softmax function, the reconstruction loss function above encourages the model to produce a higher probability on the original token while ignoring the rest of the candidates. It does not consider the other tokens that may have a similar meaning under a given context. This issue significantly limits the variety of usable alternative tokens. Additionally, this loss function relies on a single softmax function for multi-object learning; it cannot provide the expressiveness required by the language model [Yang et al., 2017]. We inspect the candidates and, in most of the cases, only the top-ranked token fits the context in the text. This is problematic because the exponential mechanism for our sampling process also relies on the other candidates to generate the text, as required by Equation 3.10.

To address the above issue we propose a novel embedding reward function using the pre-trained word embeddings. Word representation learning models [Mikolov et al., 2013] show that discrete text tokens' semantic can be embedded into a continuous latent vector space. The distance between word embedding vectors can be a reference to measure the similarity between different words. To encourage our rating function ρ_i to learn richer and better substitute tokens, we propose a reward function that leverages the semantics learned from the other corpus. The text dataset to be anonymized and released can be small, and the extra semantic knowledge learned from the other corpus can provide an additional reference for our rating function.

This reward function is inspired by the Policy Gradient loss function proposed by Sutton et al. [2000]:

$$\mathcal{L}_{embed} = -\sum_{x_i \in x, x \in \mathbb{D}} \Big(\sum_{v \in \mathbb{E}_k(\tilde{x}_i)} \log(Pr[\tilde{x}_i = v]) \gamma(x_i, v) + \sum_{w \sim \mathbb{V}_k} \log(Pr[\tilde{x}_i = w]) \gamma(x_i, w) \Big) \tag{3.13}$$

Generally, this reward function assigns credits to the under-rated tokens in the reconstruction loss function. Recall that $\mathbb{D}$ is the original dataset and x is one of its texts. At time step i, this reward function first assigns rewards to the top-k selected tokens, denoted as $\mathbb{E}_k(\tilde{x}_i)$, according to probability estimates for random variable $\tilde{x}_i$ in Equation 3.7. The rewards are proportional to their semantic relationship to the original token x_i. It is defined as a function $\gamma : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$

$$\gamma(w, v) = \min \big(\text{cosine}(Em(w), Em(v)), 0.85 \big) \tag{3.14}$$

The *min* function avoids the generator only focusing on the original token. By assigning rewards to $\mathbb{E}_k(\tilde{x}_i)$ it encourages the other candidates also having a close semantic relationship to the targeted one. However, it would fail to reach less frequent tokens. Therefore, in the second part of the reward function, we encourage the model to explore less frequent tokens by random sampling candidates as $\mathbb{V}_k$.

This design can be interpreted as balancing the exploitation (top-k) and the exploration ($\mathbb{V}_k$) in reinforcement learning [Sutton et al., 1998].

During training, the model will first be pre-trained by minimizing the reconstruction loss in Equation 3.12 through the *Adam* optimizer and adopting the embedding reward loss later. Then, the total loss is:

$$\mathcal{L} = \lambda_{recon} \times \mathcal{L}_{recon} + \lambda_{embed} \times \mathcal{L}_{embed} \tag{3.15}$$

Specifically, the reconstruction loss can lead the model to generate a grammatically correct text, and the embedding reward loss encourages the model to focus more on semantically similar tokens. To better fine-tune the model, the balance of the two loss functions are controlled by λ_{recon} and λ_{embed}.

3.4.4 Asymptotic Lower Bound of ϵ for Utility

Following the Theorem 3.6 in Weggenmann and Kerschbaum [2018], we also need a large ϵ to produce meaningful results while the discrete output space of the exponential mechanism is large.

Theorem 2. ***Necessary Condition on*** ϵ**.** *Given rating function* ρ*:* $\mathcal{M} \times \mathcal{N} \rightarrow \mathbb{R}$*, global sensitivity* Δ *and local sensitivity* $\overline{\Delta}$*. For any* $m \in \mathcal{M}$*,* $\tau \in [min(\rho(m,.)),\ max(\rho(m,.))]$ *where* $max(\rho(m,.)) = max(\rho(m,n))$ for $n \in \mathcal{N}$*, and* $min(\rho(m,.))$ *stands for the*

minimum one. Now, split $\mathcal{N}$ into $\mathcal{B}$ and $\overline{\mathcal{B}}$, where $\mathcal{B}$ stands for the set of outputs with score larger than τ and $\overline{\mathcal{B}}$ contains other outputs in $\mathcal{N}$, a probability $p \in [0,1]$, we have the necessary condition on ϵ for $Pr\left[\varepsilon_{\epsilon,\rho}(m) \in \mathcal{B}\right] \geq p$, $|.|$ denotes the input size:

$$\epsilon \geq 2\Delta/(\overline{\Delta} \ln\left(p/(1-p) * |\overline{\mathcal{B}}|/|\mathcal{B}|\right))$$

Proof.

$$\begin{aligned}
&Let\ \rho_{max} = max(\rho(m,.)) \\
&Let\ Pr\left[\varepsilon_{\epsilon,\rho}(m) \in \mathcal{B}\right] = \frac{\sum_{n\in\mathcal{B}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))}{\sum_{n\in\mathcal{N}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))} \\
&= \frac{\sum_{n\in\mathcal{B}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))}{\sum_{n\in\mathcal{B}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n)) + \sum_{n\in\overline{\mathcal{B}}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))} \\
&= \frac{1}{1 + \sum_{n\in\overline{\mathcal{B}}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))/\sum_{n\in\mathcal{B}} \exp(\frac{\epsilon}{2\Delta}\rho(m,n))} \\
&\leq \frac{1}{1 + |\overline{\mathcal{B}}| \exp(\frac{\epsilon}{2\Delta}(\rho_{max} - \overline{\Delta}))/(|\mathcal{B}| \exp(\frac{\epsilon}{2\Delta}\rho_{max}))} \\
&= |\mathcal{B}|/(|\mathcal{B}| + |\overline{\mathcal{B}}| \exp\left(-\epsilon\overline{\Delta}/2\Delta\right)) \geq p, \\
&We\ get\ \epsilon \geq 2\Delta/\overline{\Delta} \ln\left(p/(1-p) * |\overline{\mathcal{B}}|/|\mathcal{B}|\right).
\end{aligned}$$

□

Since $\overline{\Delta} \leq \Delta$, when $p = 1/2$:

$$\epsilon \geq 2\ln\left(\frac{p}{1-p} * \frac{|\overline{\mathcal{B}}|}{|\mathcal{B}|}\right) = 2\ln\left(\frac{|\overline{\mathcal{B}}|}{|\mathcal{B}|}\right) = 2\ln\left(\frac{|\mathcal{N}| - |\mathcal{B}|}{|\mathcal{B}|}\right).$$

Usually, for textual data, the size of $\mathcal{B}$, which contains meaningful token candidates, is small while choosing meaningful outputs with an acceptable τ. According

to the above equations, we have $\epsilon \in \Omega(\ln(|\mathcal{N}|))$ to get useful results, where Ω means the asymptotic lower bound. For instance, if the vocabulary size is 20,000, then the asymptotic lower bound for ϵ is $\Omega(9.9)$ to produce a meaningful result.

Algorithm 1 Training Procedure of Embedding Reward Auto-Encoder

INPUT: $Dataset : \mathbb{D}$, $learning\ rate : lr$.
PARAMETERS: θ.
COMPONENTS: $Encoder : E_\theta()$, $Generator : G_\theta()$.

Initialize θ.
for N epochs **do**
  for Text x **in** $\mathbb{D}$ **do**
    Produce the latent vector: $E_\theta(x)$.
    Probabilities of new tokens: $Pr[\tilde{x}] \leftarrow G_\theta(E_\theta(x))$.
    Reconstruction loss: $\mathcal{L}_{recon} \leftarrow \sum_{x_i \in x} -\log Pr\left[\tilde{x}_i = x_i\right]$.
    Embedding reward loss:
      TOP k: $\mathcal{L}_{embed} \leftarrow -\sum_{x_i \in x} \Big(\sum_{v \in \mathbb{E}_k(\tilde{x}_i)} \log(Pr[\tilde{x}_i = v])\gamma(x_i, v)\Big)$.
      Random K: $\mathcal{L}_{embed} \leftarrow -\sum_{x_i \in x} \Big(\sum_{w \sim \mathbb{V}_k} \log(Pr[\tilde{x}_i = w])\gamma(x_i, w)\Big)$
    Total loss: $\mathcal{L} \leftarrow \lambda_{recon} \times \mathcal{L}_{recon} + \lambda_{embed} \times \mathcal{L}_{embed}$.
    Update parameters:
      $\theta \leftarrow \theta - lr \nabla \mathcal{L}$.
  end for
end for

OUTPUT: θ

3.5 Experiment

All the experiments are carried out on a Windows Server equipped with two Xeon E5-2697 CPUs (36 cores), 384 GB of RAM, and four NVIDIA TITAN XP GPU cards. We evaluate ER-AE on two different datasets with respect to its effectiveness for privacy protection and utility preservation.

Algorithm 2 Evaluation Procedure of Embedding Reward Auto-Encoder

INPUT: $Text : x$.
PARAMETERS: θ.
COMPONENTS: $Encoder : E_\theta()$, $Generator : G_\theta()$.

Load trained parameters: θ.
Produce the latent vector: $E_\theta(x)$.
Probabilities of new tokens: $Pr[\tilde{x}] \leftarrow G_\theta(E_\theta(x))$.
for x_i **in** x **do**
  Apply exponential mechanism and get: $Pr[\varepsilon_{\epsilon,\rho_i}(\tilde{x}_i)]$.
  Sample tokens: $\tilde{x}_{dp}[i] \sim \varepsilon_{\epsilon,\rho_i}(\tilde{x}_i)$.
end for

OUTPUT: $Newtext : \tilde{x}_{dp}$

Table 3–1: Information of Datasets

Dataset	#reviews	#sentences	#authors
Yelp Review Dataset	76,241	200,940	100
Academic Review Dataset	17,719	268,253	N/A

3.5.1 Datasets

- **Yelp Review Dataset**[1] : All the reviews and tips that come from the top 100 reviewers ranked by the number of published reviews and tips. It contains 76,241 reviews and 200,940 sentences written by 100 authors.

[1] http://www.yelp.com/dataset_challenge

- **Academic Review Dataset**: All the public reviews from NeurIPS (2013-2018) and ICLR (2017) based on the original data and the web crawler provided by Kang et al. [2018]. It has 17,719 reviews, 268,253 sentences, and the authorship of reviews is unknown.

Each dataset is divided into 70/10/20 for train/dev/evaluation, respectively.

3.5.2 Preprocessing

All the reviews are first tokenized into sentences through the API of *NLTK* python package[2] . Then, each sentence is tokenized into a vector of tokens using the same package. The sentences that are longer than the maximum length are cut into the maximum length, and the shorter ones are padded with zero. During training the input sentences of the generator are inserted a start token at the sentence beginning. The target sentences of the generator end with an end token. We also adopt the word drop technology with a probability of 0.5 to avoid exposure bias problem during the training of sequence generation.

3.5.3 Baselines

As mentioned in Chapter 2, most of the controllable text generation and style transferal studies rely on known authorship or other labels [Shetty et al., 2018, Hu et al., 2017]. They are not applicable to our problem. Therefore, we pick SynTF [Weggenmann and Kerschbaum, 2018] and different generation and sampling models for evaluation:

[2] https://www.nltk.org/

- **Random Replacement (Random-R)**: This method generates a new text by replacing each token in the text by randomly picking substitution from the vocabulary.
- **Auto-encoder (AE)**: A bidirectional auto-encoder trained using the reconstruction loss in Equation 3.12.
- **AE with Differential Privacy (AE-DP)**: Extended version of AE with the added exponential mechanism for text generation. It does not include the embedding reward from Equation 3.13.
- **SynTF [Weggenmann and Kerschbaum, 2018]**: We directly generate the tokens through SynTF's differentially-private sampling function without further extraction of the frequency vector.

3.5.4 Experiment Setting

For ER-AE, we adopted a two-layers stacked GRU network for both the encoder and the generator. There are 512 cells in each GRU layer. The vocabulary size in all experiments is 20,000, separately built for each dataset. All the word embeddings in our model come from the pre-trained *BERT* embeddings provided by [Devlin et al., 2019], which has a dimension of 768 for each embedding. The maximum input length of our model is 50, the learning rate is 0.001, the k for embedding reward loss function is 5, the λ_{recon} is 1, the λ_{embed} is 0.0 at the beginning, which is increased to 0.5 after the first epoch, and the batch size is 128. ER-AE is implemented in python through TensorFlow [Abadi et al., 2016a].

3.5.5 Evaluation Metrics

All the models are evaluated from three aspects: semantic preservation, privacy protection, and stylometric changes:

- **Semantic Preservation (USE)**: A pre-trained *Universal Sentence Embedding Similarity* (*USE*) model[3] from Google. It can embed a sentence into a latent vector that represents its semantics [Cer et al., 2018]. It is widely used for supervised NLP tasks such as sentiment analysis [Li et al., 2018]. We measure the degree of semantic preservation using the cosine similarity between the latent vector of the original text and one of the generated text.
- **Privacy Protection (Authorship)**: One of the state-of-the-art authorship identification neural network models [Sari et al., 2017] is adopted to identify the authorship of generated text. The model is first trained on the training dataset, and the performance is evaluated on the testing set. An author's privacy is protected if s/he cannot be identified using authorship identification techniques.
- **Stylometric Changes**: Well-established stylistic context-free features such as text length and a number of function words. We adopt *StyloMatrix* [Ding et al., 2017] for an aggregation of features in Iqbal et al. [2013], Zheng et al. [2006]. The feature vector change before/after generation is measured by the difference in L2 norm.

[3] https://tfhub.dev/google/nnlm-en-dim128-with-normalization/1

3.5.6 Results

In this section, we demonstrate the performance of our model through quantitative evaluation and case study.

Quantitative Evaluation

The result of quantitative evaluation is shown in Table 3–2 and Table 3–3. With a USE score around 0.2 for both the Yelp review dataset and the academic review dataset, SynTF and Random-R generate grammatically incorrect text and completely change the meaning of the original one. Compared to SynTF and Random-R, the texts generated by ER-AE achieve a significantly higher utility score, over 0.79 for Yelp reviews and 0.74 for academic reviews. SynTF and Random-R perform better on authorship obfuscation and stylometric changes due to the fact that the generated texts are almost irrelevant to the original.

AE and AE-DP generate texts that have a high utility score around 0.9, but the chance of a successful authorship identification attack is also high. With 100 candidate authors in the Yelp dataset, the authorship identification model achieves about 20% accuracy, which only drops by around half compared to the original 55% identification risk. The generated texts are very close to the original. In contrast, ER-AE achieves a significantly lower authorship identification score of 7%, which is significantly lower than the original identification risk. It indicates that ER-AE hides an author's writing style much better than AE and AE-DP under the same settings.

Impact of Embedding Reward

We look into the top five candidates when the generator samples a token for our model and the original auto-encoder. Table 3–5 shows that the embedding reward

Table 3–2: Results for each evaluation metric on Yelp datasets. ↑ indicates the higher the better. ↓ indicates the lower the better.

Model	Yelp (100-author)		
	USE ↑	Authorship ↓	Stylometric↑
Original text	1	0.5513	0
SynTF [Weggenmann and Kerschbaum, 2018]	0.1955	0.0518	26.3031
Random-R	0.1183	0.0188	62.99
AE	0.9001	0.2312	5.5609
AE-DP	0.8966	0.1586	5.12
ER-AE (ours)	0.7963	0.0713	12.49

Table 3–3: Results for USE and Stylometric change metrics on conferences' dataset. ↑ indicates the higher the better. ↓ indicates the lower the better.

Model	Conferences' Dataset	
	USE↑	Stylometric↑
Original text	1	0
SynTF [Weggenmann and Kerschbaum, 2018]	0.2161	25.95
Random-R	0.1356	65.624
AE	0.9114	4.339
AE-DP	0.8712	5.8389
ER-AE (ours)	0.7448	10.18

Table 3–4: The intermediate result of top five words and their probabilities at the third and the fourth generation time steps

Input: there are **several unique** hot dog entrees to choose ...		
	several	**unique**
AE-DP	**several 0.98**, those 0.007, some 0.003, various 0.002, another 0.001	**unique 0.99**, different 0.0, new 3.1e-05, nice 2.5e-05, other 2.1e-05
ER-AE	many 0.55, some 0.20, **several 0.14**, different 0.04, numerous 0.03	**unique 0.37**, great 0.21, amazing 0.15, wonderful 0.1, delicious 0.05

plays an important role in selecting semantically similar candidates for substitution. AE assigns a large probability to the original token and a tiny probability to the others. If applied with the exponential mechanism, it needs a small ϵ value to sample the other tokens than the original. ER-AE shows a smoother distribution on the vocabulary and assigns higher probabilities to several top-ranked semantically relevant tokens. Its generated candidates are better. By having this smoother distribution, the exponential mechanism performs better on token substitution with a smoother distribution and significantly higher probabilities on semantically similar tokens; the model is able to preserve the semantics of the input text.

Case Study

Table 3–5 shows that both SynTF and Random-R cannot generate human-friendly text. Due to the issue of reconstruction loss function [3.12], AE and AE-DP generate samples that rarely change the original. ER-AE, powered by embedding reward, can substitute some tokens with semantically similar ones: "asian" is replaced by "portuguese", and the whole sentence still makes sense. In addition, it can preserve the grammatical structure of the input text. However, due to some missing information from word embeddings, the model fails to generate good candidates for sampling. The last sample replaces "reduce" with "dig", which changes the semantics of the input.

Utility vs. Privacy

The relative performances of semantic preservation (USE Similarity), authorship obscuration (Authorship Error rate), and writing style change (Stylometric L2 distance) are used to show the trade-off between privacy and utility. In Figure 3–7,

Table 3–5: Sample Texts Generated by Models

Input	maybe , i just missed this point in the proof.
SynTF	cayman stunts accounts pierced nickel mai aisles maddox possesses ...
Random-R	establishments morning intercepted fragrance penny ...
AE	maybe , i just missed this point in the proof .
AE-DP	maybe , i just missed this point in the proof .
ER-AE	maybe , i just missed this restoration in the proof .
Input	the novelty is combining several known ideas , which is perfectly acceptable .
SynTF	sherwood few mats confronts biceps shuffled whereby magical confirming...
Random-R	coulter twice illuminating affair bavarian schooling...
AE	the novelty is combining several known ideas , which is perfectly acceptable .
AE-DP	the novelty is combining several known ideas , which is perfectly acceptable .
ER-AE	the novelty is decoder-based coincides known ideas , which read/compose/write perfectly acceptable .
Input	attitude of the old asian lady did not help either.
AE	attitude of the old asian lady did not help either .
AE-DP	attitude of the old asian lady did not help either .
ER-AE	attitude of the old portuguese lady did not salute either .
Input	please reduce it or move some of the results to an appendix section.
AE	please reduce it or move some of the results to an appendix section.
AE-DP	please reduce it or move some of the results to an appendix section.
ER-AE	please dig it or move confounded of the results to an appendix section.

the privacy budget ϵ controls the privacy and utility of generated data. A larger ϵ means better utility but less privacy protection and vice versa. By observing those results, the optimal ϵ for the Yelp dataset is 30, and for conferences reviews the dataset is 32. The choice of an ϵ is very flexible, another ϵ value could be picked to fit the requirements of generated data.

Theoretically, an ϵ of around 30 cannot provide a strong privacy guarantee. To provide a stronger privacy guarantee typically one derives a tight local sensitivity denoted as Δ_{local} rather than using the global maximum sensitivity. Then the exponential mechanism actually provides $(\epsilon \cdot \Delta_{local})$-differential privacy. However, in our

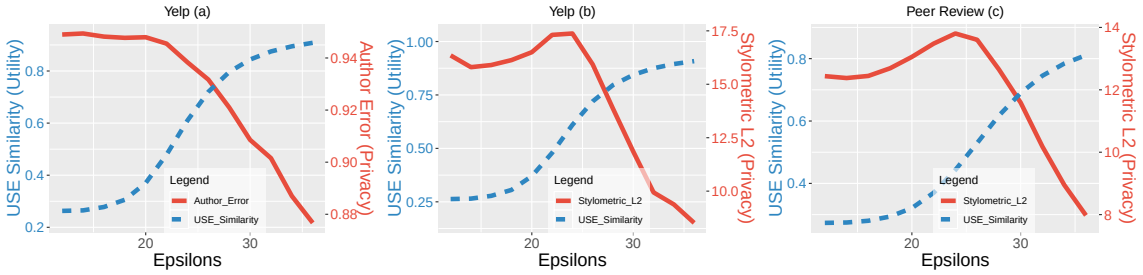

Figure 3–7: Privacy vs. Utility. Comparing USE similarity (utility), authorship identification error rate (privacy), and Stylometrics L2 distance (privacy) for different ϵs on applicable datasets. For all blue lines belonging to USE similarity metrics, the first red line is authorship identification error rate, and the last two red lines are Stylometrics L2 distance metric

.

case as well as all the existing work that builds on the concept of document indistinguishability, this tighter bound is intractable without putting a semantic limitation on the adjacency definition. Given two texts, x_1 and x_2, if they are very similar in topic, sentiment, semantics, etc., then Δ_{local} should be very small because our generator produces results based on the semantics and the generated texts $\tilde{x_{1dp}}$ and $\tilde{x_{2dp}}$ should also share a similar semantic similarity. In this case, the model provides a strong privacy guarantee because $(\epsilon \cdot \Delta_{local})$ becomes small when Δ_{local} is very small. However, when x_1 and x_2 are very different, Δ_{local} would be close to the global maximum sensitivity of 1 since they can be easily distinguished in the aspect of semantic similarity. Since it is nontrivial to prove the tight sensitivity of this model, the upper bound of sensitivity is required during the generation stage in Equation 3.10.

However, ϵ is a relative value that implies different degrees of risk given different problems [Weggenmann and Kerschbaum, 2018, Fernandes et al., 2018]. Empirically, we show that an ϵ of 30 is already enough to significantly reduce the chance of a

successful authorship identification attack. Theoretically, we offer two reasons that a large ϵ is necessary to produce meaningful texts:

First, by analyzing the privacy properties under the recently used concept of document distinguishability [Fernandes et al., 2018], we observe that a large ϵ is necessary to allow the generated text can be distinguished by the semantic information such as topic and sentiment under the strictest adjacency definition. ϵ bounds the impact of an individual has on a query result, however, usually, it does not directly relate to the disclosure of the individual's information [Lee and Clifton, 2011, Clifton and Tassa, 2013]. Text indistinguishability considers all aspects of a text including the topic, sentiment, semantics, and authorship, etc. For instance, for two completely indistinguishable texts, one cannot distinguish their authorship, sentiment, and even topics. They all look random. This case corresponds to the scenario when the user sets a very low ϵ. If the ϵ is set to be very high, it means that the two adjacent texts are very different in all aspects. The privacy budget ϵ controls the degree of such indistinguishability. However, as ϵ decreases, our model can change writing style information first and try to retain as much as semantic information. This is achieved by the REINFORCE training. It is able to arrive at a point where only the authorship identity is indistinguishable and other aspects are kept (i.e. topic, sentiment, semantics, etc.) at best. Therefore, a large ϵ is necessary to allow the generated text can be distinguished by the semantic information such as topic and sentiment under the strictest adjacency definition. However, this larger ϵ does not mean that the author's identity is also distinguishable. As mentioned earlier, a large

ϵ of 30 can already protect an author's identity by reducing the chance of a successful authorship identification attack from 55% to 7%.

Second, theoretically, a large ϵ is necessary for the exponential mechanism to produce meaningful text with a large discrete output space. As shown in the proof of Theorem 2, a higher ϵ is intrinsically necessary for a large discrete space, in our case the vocabulary, to generate meaningful and relevant text. The best ϵ of around 30 satisfies the proved asymptotic lower bound of $\Omega(9.9)$ for a vocabulary size of 20,000.

This large ϵ issue is well-known in recent works [Sala et al., 2011, Weggenmann and Kerschbaum, 2018, Fernandes et al., 2018]. Sala et al. [2011] utilize an ϵ of 100 for edge privacy, and Weggenmann and Kerschbaum [2018] use an ϵ of 42.5. Fernandes et al. [2018] leverage an ϵ of 30. They all still claim their methods are differentially-private.

Our model also satisfies the definition of differential privacy, though, theoretically, the privacy guarantee is weak. However, we have already significantly reduced the optimal ϵ value of 42.5 used by Weggenmann and Kerschbaum [2018] to around 30 given the same dataset. One possible way to lower the bound of ϵ is to directly factor in authorship and utility, such as topics, into the privacy model. However, it limits applicable to datasets.

3.6 Summary

In this chapter we propose a novel model and loss function to protect an individual's privacy and anonymize authorship by generating differentially-private text. To the best of our knowledge, this is the first model to apply the exponential mechanism into auto-encoder and generate differentially-private text. Though, our model

requires an ϵ of around 30, we provide two reasons that a large ϵ is necessary to produce meaningful texts. First, a large ϵ is necessary to allow the generated text can be distinguished by the semantic information under the strictest adjacency definition. Second, as shown in the proof of Theorem 2, a large ϵ is necessary for the exponential mechanism to produce meaningful text with a large discrete output space. We demonstrate the ability of our model to generate differential-private text with correct grammar and similar semantic on the Yelp dataset and NeurIPS & ICLR reviews datasets. The experiments show that our model outperforms on authorship obscuration and semantic preservation. The human-friendly text generation ability is shown by the samples in Table 3–5. By observing the intermediate results of the generator, the ability of the embedding reward loss function is proved to be efficient to help the generator assign higher probabilities to semantically similar tokens. The trade-off between utility and privacy is shown by visualizing the relative performances of USE and authorship attack under different levels of privacy budget ϵ.

CHAPTER 4
Generative Adversarial Network with Back-Translation

In this chapter, we introduce our second model, Generative Model with Back-Translation (GM-BT), for text anonymization. Given a reference dataset, GM-BT learns to transfer the writing style of input text into the writing style of the reference dataset without changing its semantics. With an auto-encoder and a discriminator, GM-BT is trained by the adversarial learning method and the back-translation loss function. The effectiveness of GM-BT is evaluated through experiments on two different reviews dataset on semantics preservation, writing style change, and authorship obscuration. We compare GM-BT with all baselines and previous proposed ER-AE through quantitative analysis and case study.

4.1 Preliminaries

4.1.1 Generative Adversarial Network

The *Generative Adversarial Network (GAN)*[Goodfellow et al., 2014] is a framework to train a generative model through an adversarial learning method, which has been widely applied in text generation tasks [Yu et al., 2017, Shetty et al., 2018, Hu et al., 2017]. A basic GAN consists of two components: a discriminator and a gen-

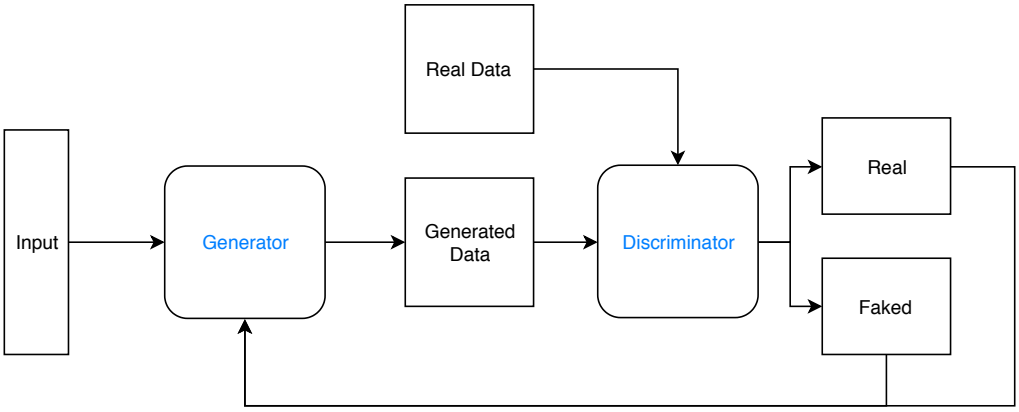

Figure 4–1: Overall architecture of GAN.

erator. The goal of GAN is to train a generator that creates fake data objects that look real. The discriminator has access to real objects and generated data objects as training data. Its mission is to learn to differentiate fake data objects from real data objects. The training process of a GAN is in an iterative way. The discriminator tries to distinguish fake objects generated by the current generator from real objects. As shown in Fig. 4–1, the discriminator receives both generated objects and real objects offers the feedback to the generator. The generator improves the generated objects based on the feedback of the new discriminator. A GAN is considered successful if the discriminator can no longer clearly differentiate whether the fake objects from the generator are real or fake.

The training process of a GAN starts with the discriminator that learns to recognize whether a data is real. At the beginning, the discriminator is easy to converge because the generator is too weak to generate good samples. To beat the discriminator, the generator tries to produce better samples than before. After repeating this for a couple of times, we get a powerful discriminator and a strong generator.

This process is called adversarial training. Formally, according to Goodfellow et al. [2014], given a discriminator D, a generator G, the distribution of real data objects $p_{\text{data}}(\boldsymbol{x})$, and a distribution of input noise variables $p_{\boldsymbol{z}}(\boldsymbol{z})$, we have the minimax value function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (4.1)$$

The discriminator wants to maximize this value function, while the generator tries to minimize it.

4.1.2 Conditional Generative Adversarial Network

The original GAN receives a vector sampled from a normal distribution, and there is no conditional constraint for generated data. However, generating data based on given information is demanded in data generation tasks. For example, one would not only want to generate human faces but also would like to control the attributes of generated faces. The conditional GAN [Mirza and Osindero, 2014] extends the ability of a discriminator. In addition to telling whether a generated sample is real or not, it can also justify whether the generated sample satisfies the given information. Therefore, to beat the discriminator the generator is required to produce the real-looking data that satisfies the given information. Then, the loss function of conditional GAN is:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]. \quad (4.2)$$

Compared with the Equation 4.1, the value function of the conditional GAN includes conditional information $\boldsymbol{y}$.

4.1.3 Gumbel-softmax

Gumbal-Softmax [Jang et al., 2017] is designed to solve the non-differentiable issue of the discrete selection operation after softmax function in GANs through estimating the non-differentiable discrete selection operation with a differentiable function.

Originally, softmax function parameterizes the multinomial distribution on one-hot-encoding outputs in terms of a continuous d-dimensional vector h as:

$$[softmax(h)]_i = \frac{\exp(h_i)}{\sum_j \exp(h_j)}, for\ i = 1 \dots d.$$

Jang et al. [2017] show that sampling result according to probability produced by the above equation is equal to the following:

$$output = one\_hot(\underset{i}{\text{argmax}}(h_i + g_i)),$$

where g_i are independent and follow a Gumbel distribution with zero location and unit scale.

Since the *one_hot* operation is non-differentiable, Gumbel-softmax approximates it through the differentiable function:

$$output = softmax(1/\tau(h + g)),$$

where τ is the inverse temperature. A lager τ would result a smoother distribution. When $\tau \to 0$, the above output would be very close to a one-hot encoding.

4.2 Problem Definition

Definition 4.2.1. Text Anonymization through Style Transfer. Let $\mathbb{D}_i$ denote a text dataset that contains a set of texts, where $x \in \mathbb{D}_i$ is one of them. Let $\mathbb{D}_t$ denote the reference dataset. Given any authorship identifier D_{ai}, for each $x \in \mathbb{D}_i$, the model transfers x into $\tilde{x}$ that has the writing style of $\mathbb{D}_t$, and D_{ai} cannot identify the real author while receiving $\tilde{x}$. The generated text $\tilde{x}$ is expected to have a close semantic similarity to the original text x. □

Definition 4.2.2. Reference Dataset. Given a text dataset $\mathbb{D}$, a reference dataset, $\mathbb{D}_t$, is expected to have a similar number of texts with $\mathbb{D}$, be written by multiple authors with a mixture of writing styles, and have a vocabulary that is mostly overlapped with the one of $\mathbb{D}$. □

The model hides the authorship of a given text and protects privacy by imitating the writing style of the reference dataset ($\mathbb{D}_t$). Given the reference dataset, in the GAN framework, the discriminator tries to distinguish whether the input text satisfies the expected writing style or not. The generator would learn how to transfer the writing style according to the feedback from the discriminator with adversarial training. For the back-translation part, the model takes input from the training set and reference dataset randomly. It "translates" the input into a random writing style and "translates" it back to the original writing style. The model learns to transfer the writing style based on the back-translation loss function.

4.3 Generative Model with Back-Translation

In this section, we provide an overview of the proposed model and explain every component in detail. Generative Model with Back-Translation (GM-BT) is a combination of conditional GAN and back-translation loss function. It contains an encoder and a decoder. In order to hide the authorship of the given text, another dataset is required to be a reference dataset. The model receives a sequence of tokens as input, and the encoder extracts content information in the input data and outputs a feature vector. The feature vector is fed into the decoder after being concatenated with a writing style embedding of the reference dataset. The decoder generates new style text based on given information. GM-BT is trained by the reconstruction loss function, the GAN loss function, and the back-translation loss function. The GAN loss function, as shown in Figure 4–2, guides the encoder and the generator through providing feedback from the discriminator based on current generated text. The back-translation loss function, as shown in Figure 4–3, firstly "translates" the input text into a text with a random writing style, then, it "translates" the text back to have the same writing style with the input text. It provides feedback through calculating the reconstruction loss between the back-translated text and the input text. The reconstruction loss function can speed up the training process and help the generator learn the grammar structure in the beginning. The back-translation loss function can not only guide the generator to preserve the semantics of input but give freedom to the generator to change words. The GAN loss function offers feedback on the quality of the generated samples to the generator.

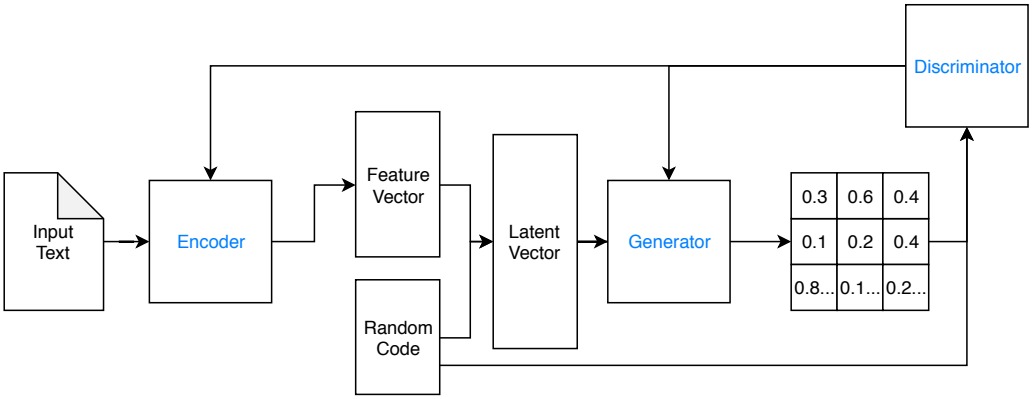

Figure 4–2: The GAN architecture of GM-BT.

However, there exists a differentiability issue while training GANs for textual data. The gradients cannot be backpropagated from the discriminator because the original generation process that contains the discrete selection is non-differentiable [Yu et al., 2017]. To make this process differentiable, we apply the *Gumbal-softmax* [Jang et al., 2017] to replace the original softmax function and discrete selection during adversarial training. The Gumbel-softmax produces soft vectors through approximating the discrete selection process with a differentiable function instead of one-hot vectors, which allows gradients to flow from the discriminator to the generator.

4.3.1 Encoder

Similar to the encoder in Chapter 3, GM-BT also includes a Bi-directional RNN with a GRU cell as the encoder. The encoder produces a feature vector $E(x)$ after receiving a text x as input that is converted to a sequence of token embeddings $\langle Em(x_1), \ldots, Em(x_l)\rangle$ after being tokenized into tokens. Unlike ER-AE that directly feeds the feature vector into the generator, GM-BT concatenates the feature vector

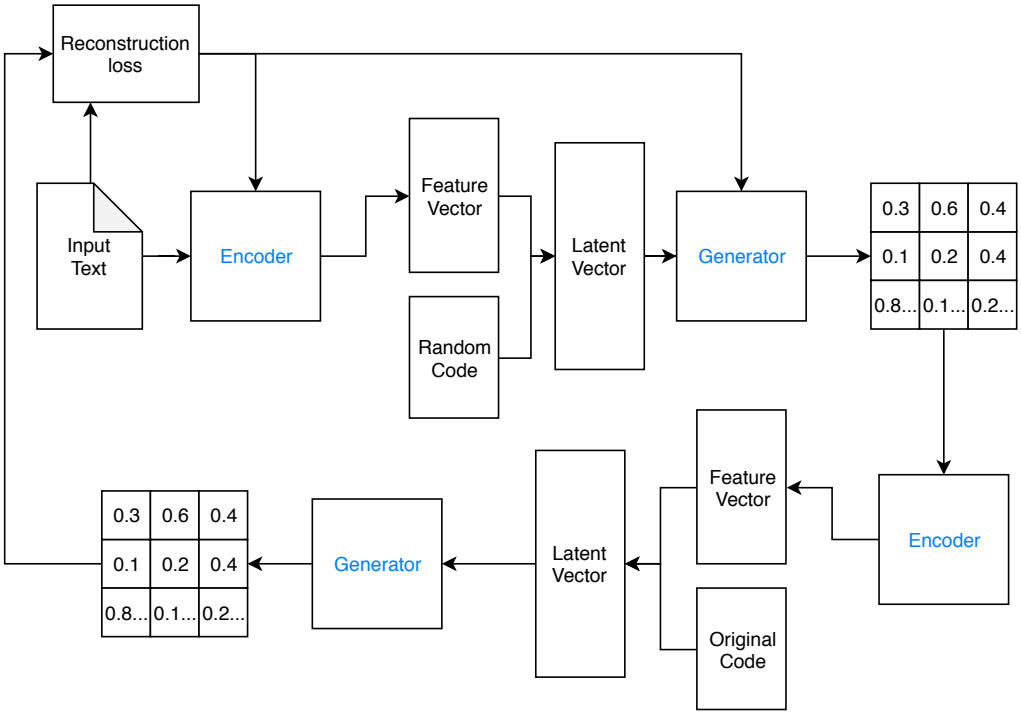

Figure 4–3: Overall architecture of the back-translation loss of GM-BT.

with a writing style embedding, the conditional code C. C is a one-hot encoding that controls the expected writing style of the produced text. For the training set writing style, C is [1, 0], for the reference dataset, C is [0, 1]. We have the final latent vector:

$$lat_{x,C} = concat(E(x), C),$$

where $concat(,)$ is the concatenation function on the last axis.

4.3.2 Generator

The generator receives the latent vector $lat_{x,C}$ that contains the context information of the original input data x and the desired writing style C. It is also a RNN with GRU, which is similar to the one in Chapter 3. In addition, we adopt an attention mechanism [Bahdanau et al., 2014] into the generator. The Bahdanau attention enables the generator to make the prediction by looking back at the relevant information in historical hidden states of the encoder. During the i-th time step, given a sequence of the encoder's hidden states S respected to x, the context vector c_i is a weighted summation of the hidden states:

$$\alpha_{ij} = \frac{\exp(a(s^g_{i-1}, S_j))}{\sum_{k-1}^{l} \exp(a(s^g_{i-1}, S_k))},$$

$$c_i = \sum_{j=1}^{l} \alpha_{ij} S_j,$$

where S is the concatenation of the forward and backward states in the encoder, α_{ij} is the weight of S_j, $a(,)$ is a feedforwad neural network model, and s^g_i is the i-th hidden state of the generator. To plug c_i into a GRU cell, instead of feeding the

embedding of input token into the cell it is concatenated with the context vector:

$$Em_{new}(x_i) = concat(Em(x_i), c_i).$$

Then, the hidden state update and token prediction are made based on the new input vector. The generator learns to produce text based on the content information $E(x)$ and conditional code C. C controls the expected style of the text produced by the generator.

4.3.3 Discriminator

The discriminator D is a *Convolutional Neural Network (CNN)* text classifier proposed by Kim [2014]. CNNs were originally designed for computer vision tasks; they play an important role in most computer vision systems. Following the success of CNNs in computer vision, current studies find that it is also powerful in natural language processing problems. Instead of using image pixels, the text can be represented as a numerical matrix. Given a text x, after tokenizing the text each token x_i is converted into a token embedding $Em(x_i)$, which is the corresponding row in the matrix.

The architecture of CNN is straightforward. A couple of convolutional layers with different kernel sizes extract various feature vectors $\langle vec_1, \ldots, vec_i \rangle$ from the input data x. Then, a max pooling layer with output size 1 is applied on each feature vector. All the results produced by the pooling layer are concatenated as the final feature vector vec_{final}. The prediction is made through a fully connected layer

based on the feature vector:

$$prediction = Softmax(\boldsymbol{W}_f \times vec_{final}),$$

where $\boldsymbol{W}_f \in \mathbb{R}^{nclass \times vf}$ are the weights in the fully connected layer, $nclass$ is the number of classes, and vf is the dimension of the final feature vector.

The input to the discriminator is the Gumbel-softmax output of current generator and the randomly sampled text from the reference dataset and the training set. The one-hot-encoding writing style code C can be treated as the target label. The discriminator tries to distinguish whether the input text satisfies the desired writing style code C (label). This is a widely used setting in conditional GANs [Mirza and Osindero, 2014, Hu et al., 2017, Shetty et al., 2018]. The discriminator is trained through maximizing:

$$\max_D V(D,G) = \mathbb{E}_{x \sim p_{\text{ data }}(x)}[\log D(x|C) + \log(1 - D(G_{gb}(lat_{x,C}|C)))], \tag{4.3}$$

where $G_{gb}(.)$ is the function of the generator with Gumbel-Softmax that is to avoid the non-differentiability issue introduced in Section 4.3.

4.3.4 Back-Translation Loss Function

Reconstruction loss function (Equation 3.12) is a prevalent loss function in the text generation problem. By learning to reconstruct the original input text a generator learns to generate text with correct grammatical structure and keep the semantics. However, only optimizing the original input tokens strongly limits the generator on exploring other expressions for the same semantics in the style transfer problem. The model tends to generate the exact same tokens as the input rather than other

options. Therefore, the generator demands a weaker loss function that allows it to produce different text. Subramanian et al. [2018] propose the back-translation loss to control the attributes in the generated text. Given the input text x and a random code C_r, we have the generated text $\tilde{x}$ and x^{bt}, which is the generated text with the input of $\tilde{x}$ and x's writing style code C_x, then the loss function can be formed as:

$$\mathcal{L}_{bt} = \lambda_{recon} \sum_{x_i \in x} -\log Pr\left[\tilde{x}_i = x_i | C_r\right] + \lambda_{bt} \sum_{x_i \in x} -\log Pr\left[x_i^{bt} = x_i | C_x\right]. \tag{4.4}$$

The input x is first "translated" into $\tilde{x}$, and then $\tilde{x}$ is "translated" into x^{bt}. The loss function is called back-translation because x^{bt} is expected to be the same as x when the conditional code C is C_x. Under this condition we can use the reconstruction loss between x^{bt} and x to train the generator, which is weaker than the original reconstruction loss because $\tilde{x}$ does not have direct constraint when λ_{recon} is zero. λ_{recon} linearly decreases from 1 to 0 after a couple of epochs to release the power of reconstruction loss, whereas λ_{bt} is always 1.

4.3.5 Adversarial Training

Adversarial training is a vital part of a GAN framework. Proved by Yu et al. [2017], the pre-training of the generator with reconstruction loss function is necessary to speed up the whole training progress. Therefore, as shown in Algorithm 3, our model is first pre-trained for one epoch. Then, the discriminator is trained based on the current generator and the real data. After that, the generator learns to beat the discriminator while minimizing the back-translation loss function. This is repeated until the generator is convergent.

Algorithm 3 The Training Procedures of GM-BT

INPUT: $Dataset : \mathbb{D}$
PARAMETERS: θ
COMPONENTS: $Encoder : E_\theta()$, $Generator : G_\theta()$., $Discriminator : D_\theta()$.

Initialize θ.
Pre-training using reconstruction loss $\mathcal{L}_{recon}$.
while $E_\theta()$ and $G_\theta()$ do not converge **do**
  Update the Discriminator:
  for N_d times **do**
    Generative faked samples based on current generator.
    Update $D_\theta()$ through maximizing Equation 4.3.
  end for

  Update the Encoder and the Generator:
  for N_{GAN} times **do**
    Update $E_\theta()$ and $G_\theta()$ through minimizing the combination of the conditional GAN loss function Equation 4.2 and back-translation loss function Equation 4.4.
  end for
  Update the hyper-parameter λ_{recon}.
end while
OUTPUT θ

4.4 Experiment

In this section, we present the details of the conducted experiments. The platform of all experiments is the same as the one in Chapter 3.

4.4.1 Datasets and Baselines

In the experiments, we also adopt the Yelp Review dataset and the Peer Review Dataset described in Chapter 3 and Table 3–1. The baselines and evaluation metrics are also the same. We further compared GM-BT with the previously proposed ER-AE model.

4.4.2 Experiment Setting

Both the encoder and the decoder utilize a one layer GRU network. Each GRU layer has 512 cells. We built vocabulary for each dataset separately with a size of 20,000. All the token embeddings come from the pre-trained BERT model, which has a dimension of 768. Our model has a maximum input length of 50, a learning rate of 0.001, λ_{recon} of 1 that is reduced to 0 after 5 epochs, λ_{bt} of 1, N_d of 1, N_{GAN} of 5, and a batch size of 128. During the training, the temperature of the Gumbel-softmax is linearly annealed, from 1.0 for iterations 1 to 19,000 and then kept at 0.5 until training ends. The reference dataset is a subset of Wikipedia dataset[1] because it has a mixture of writing style and written by different authors. We created the subset with a similar size to our training set through randomly sampling texts whose tokens are mostly included in the vocabulary of the training set. During training, reference

[1] https://en.wikipedia.org/wiki/Wikipedia:Database_download

Table 4–1: Results for each evaluation metric on Yelp datasets. ↑ indicates the higher the better. ↓ indicates the lower the better.

Model	Yelp (100-author)		
	USE ↑	Authorship ↓	Stylometric↑
Original text	1	0.5513	0
SynTF [Weggenmann and Kerschbaum, 2018]	0.1955	0.0518	26.3031
Random-R	0.1183	0.0188	62.99
AE	0.9001	0.2312	5.5609
AE-DP	0.8966	0.1586	5.12
ER-AE (ours)	0.7963	0.0713	12.49
GM-BT (ours)	0.814	0.0634	10.06

Table 4–2: Results for USE and Stylometric change metrics on conferences' dataset. ↑ indicates the higher the better. ↓ indicates the lower the better.

Model	Conferences' Dataset	
	USE↑	Stylometric↑
Original text	1	0
SynTF [Weggenmann and Kerschbaum, 2018]	0.2161	25.95
Random-R	0.1356	65.624
AE	0.9114	4.339
AE-DP	0.8712	5.8389
ER-AE (ours)	0.7448	10.18
GM-BT (ours)	0.831	13.72

texts are randomly sampled from the reference dataset. This model is implemented in Python with TensorFlow python library. The sentence tokenizer and word tokenizer come from NLTK Python library.

4.4.3 Results

In this section, we present the results of our experiments and compare them with all baselines and ER-AE in three aspects, namely semantic preservation, privacy protection, and writing style changes.

Quantitative Evaluation

Quantitative results are shown in Table 4–1 and Table 4–2. As described in Chapter 3, with a USE score of around 0.2, SynTF and Random-R cannot generate grammatically correct texts. Our proposed model GM-BT achieves a phenomenally higher USE score of above 0.81 on both datasets, suggesting that GM-BT has a significantly better utility. Although SynTF and Random-R can better obfuscate the authorship and change the writing style, they sacrifice too much text structure and semantics. As a result, the generated text is not even a proper text.

On the other hand, AE and AE-DP have good utility, but the generated text is too easily attacked. Compared with them, GM-BT achieves a competitive utility performance on both datasets while significantly reducing the identification risk of authorship from 55% to 6%. In addition, the changes of writing style on both datasets are much higher than the two baselines. It reveals that GM-BT can protect authorship privacy much better than AE and AE-DP while performing competitively on utility.

Comparing with previously proposed ER-AE, GM-BT performs better on data utility on both datasets. On the conferences' dataset, the USE score of GM-BT is higher than the one of ER-AE with almost 0.1. At the same time, GM-BT achieves similar performance on privacy protection and writing style change with ER-AE. GM-BT reduces the identification risk from 55% to 6%, which is even better than ER-AE. This result indicates that GM-BT performs better than ER-AE on data utility with competitive performance on privacy protection and writing style change.

Table 4–3: Sample Texts Generated by Models

Input	maybe , i just missed this point in the proof.
SynTF	cayman stunts accounts pierced nickel mai aisles maddox possesses ...
Random-R	establishments morning intercepted fragrance penny ...
AE	maybe , i just missed this point in the proof .
AE-DP	maybe , i just missed this point in the proof .
ER-AE	maybe , i just missed this restoration in the proof .
GM-BT	maybe , i just missed this point in the proof .
Input	the novelty is combining several known ideas , which is perfectly acceptable .
SynTF	sherwood few mats confronts biceps shuffled whereby magical confirming...
Random-R	coulter twice illuminating affair bavarian schooling...
AE	the novelty is combining several known ideas , which is perfectly acceptable .
AE-DP	the novelty is combining several known ideas , which is perfectly acceptable .
ER-AE	the novelty is decoder-based coincides known ideas , which read/compose/write perfectly acceptable .
GM-BT	the novelty is combining several ideas , which is perfectly acceptable known .
Input	attitude of the old asian lady did not help either.
AE	attitude of the old asian lady did not help either .
AE-DP	attitude of the old asian lady did not help either .
ER-AE	attitude of the old portuguese lady did not salute either .
GM-BT	copies of the old asian did not have earned either .
Input	please reduce it or move some of the results to an appendix section.
AE	please reduce it or move some of the results to an appendix section.
AE-DP	please reduce it or move some of the results to an appendix section.
ER-AE	please dig it or move confounded of the results to an appendix section .
GM-BT	reduce it or please add some of the results to an appendix section .

Case Study

In Table 4–3 we show the generated text of models on four input texts. From the first two texts we can see that SynTF and Random-R fail to generate readable text. There is no cohesion between generated words, whereas AE and AE-DP can make little change in generated texts. They just produce the exact same texts that have high identification risk. Basically, ER-AE can preserve the content of the input

data and change some words in the text. However, the model fails to pick a good substitution word because of the limitation of word embeddings. In the last sample, ER-AE changes the semantic by replacing "reduce" with "dig". GM-BT does not have this issue; it changes the writing style by reordering words and substituting tokens if necessary. In the second and last samples, GM-BT changes the position of "known" and "please" to change the writing style. In the last text it replaces "move" with "add". According to the generated samples, GM-BT has a higher chance to produce a fluent and grammatically correct text. GM-BT also has weaknesses, such as the first sample, when the text is too short it would fail to change the writing style.

GM-BT vs. ER-AE

Generally, GM-BT performs better than ER-AE on USE scores, which means the generated data has better utility than that of ER-AE. In terms of privacy, GM-BT also achieves a significantly lower identification risk than ER-AE on the authorship attack and a competitive change on the writing style. We also carefully examine the generated samples from both models. Due to the sampling processing, ER-AE is more likely to generate some irrelevant vocabularies for substitutions. In contrast, GM-BT does not have a sampling operation in the generator, and therefore it has a higher chance to generate a human-friendly text that preserves the semantics of the input data.

4.5 Summary

In this chapter we propose the second novel method to imitate the writing style of a reference dataset to protect the privacy of authorship. To the best of our knowledge,

this is the first model to anonymize authorship by automatically imitating a writing style. The attention mechanism is utilized to improve the power of the generator. We avoid the limitation of the reconstruction loss function by replacing it with back-translation loss. Combining the back-translation loss function with GAN loss, the model can be trained end-to-end, which means it is easy to train. We evaluate our model and show a competitive ability to generate text that preserves the semantics while changing the writing style on the Yelp dataset and NeurIPS & ICLR reviews datasets. By showing samples generated from all models, we observe that GM-BT generally performs better than ER-AE on text fluency and grammatic correctness.

CHAPTER 5
Conclusions & Future Work

In this thesis we propose two novel models, Embedding Reward Auto-Encode (ER-AE) and Generative Model with Back-Translation (GM-BT), to tackle the authorship anonymization problem while releasing textual data. ER-AE extends the auto-encoder with an exponential mechanism to generate differentially-private text. Though the privacy guarantee is not strong, which is due to the lack of provable tight sensitivity bound, we give two reasons that a large ϵ is necessary to produce meaningful texts. First, a large ϵ is necessary to achieve only the author's identity indistinguishability while allowing the generated text can be distinguished by the semantic information under the strictest adjacency definition. Second, we theoretically prove that a large ϵ is necessary for the exponential mechanism to produce meaningful and relevant text with a large discrete output space. To train the model and overcome the limitation of the exponential mechanism, the embedding reward loss function is designed to improve the performance of the generator. GM-BT is generally a GAN model, which utilizes an attention generator. During training it combines the back-translation loss with GAN loss. We evaluate both models using real-life datasets. The results show that both models can generate human-friendly text with correct grammar while preserving the semantic of the input and protecting

an individual's privacy. They have different strengths and weaknesses. ER-AE would generate texts that may contain grammatical mistakes due to the sampling operation in the exponential mechanism. Whereas, by avoiding the sampling operation, GM-BT usually generates a better understood text. However, its privacy protection is not guaranteed. Both models outperform on the quantitative experiments and show the ability to defend an authorship identification attack through changing the writing style.

We find that ER-AE performs well on short sentences but cannot handle long texts well. Also, the grammar structure is not always correct. Our future research will focus on its performance on long sentences, paragraphs, and correct grammatical structure. The following are some potential directions for improving ER-AE:

- The embedding reward loss function is well defined but not perfect. It can increase the probability of some semantically similar words, but it has a problem for multi-object learning. It can be further improved by adopting loss functions from that area.
- The exploration part of the embedding reward loss function is in an intuitive way; the way to choose tokens can also be investigated.
- New differentially-private mechanisms can be further studied since most of the bad samples generated by ER-AE are caused by the non-zero items produced by the exponential mechanism. A mechanism that can produce sparse distribution would significantly improve the performance of ER-AE.

- ER-AE does not perform well on long texts. The recent proposed *Transformer* [Vaswani et al., 2017], which can better process long-term information in sequence data, would be a good direction for further exploration.

On the other hand, GM-BT could produce a better grammatically correct text than ER-AE, but the writing style change on short texts is not significant. The following are some potential directions for improving GM-BT:

- Design a better loss function to replace the back-translation loss. Although it is weaker than the reconstruction loss function, a new way to learn the grammatical structure and content preservation is desired to encourage the model to make more changes on generated texts.
- Find a way to encourage the generator to make changes in a generated text. Reinforcement learning method would be a good choice to provide more types of feedback on the quality of generated text, e.g., the change of the text structure reward and the change of words reward.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16*, pages 265–283, 2016a.

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016b.

Balamurugan Anandan, Chris Clifton, Wei Jiang, Mummoorthy Murugesan, Pedro Pastrana-Camacho, and Luo Si. t-plausibility: Generalizing words to desensitize text. *Trans. Data Privacy*, 5(3):505–534, 2012.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian

Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://www.aclweb.org/anthology/D18-2029.

Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4666–4677, 2018.

Rui Chen, Benjamin CM Fung, Philip S Yu, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB JournalThe International Journal on Very Large Data Bases*, 23(4):653–676, 2014.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103, 2014a.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014b.

Chi-Yin Chow and Mohamed F Mokbel. Trajectory privacy in location-based services and data publication. *ACM SIGKDD Explorations Newsletter*, 13(1):19–29, 2011.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. In *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, pages 88–93. IEEE, 2013.

Jorge Cortés, Geir E Dullerud, Shuo Han, Jerome Le Ny, Sayan Mitra, and George J Pappas. Differential privacy in control and network systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4252–4272. IEEE, 2016.

Fida Kamal Dankar and Khaled El Emam. The application of differential privacy to health data. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 158–166. ACM, 2012.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Steven HH Ding, Benjamin CM Fung, and Mourad Debbabi. A visualizable evidence-driven approach for authorship attribution. *ACM Transactions on Information and System Security (TISSEC)*, 17(3):12, 2015.

Steven HH Ding, Benjamin CM Fung, Farkhund Iqbal, and William K Cheung. Learning stylometric representations for authorship analysis. *IEEE transactions on cybernetics*, (99):1–15, 2017.

Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489. ACM, 2018.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Natasha Fernandes, Mark Dras, and Annabelle McIver. Author obfuscation using generalised differential privacy. *arXiv preprint arXiv:1805.08866*, 2018.

Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets.

In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596. JMLR. org, 2017.

Farkhund Iqbal, Hamad Binsalleeh, Benjamin CM Fung, and Mourad Debbabi. A unified data mining solution for authorship analysis in anonymous textual communications. *Information Sciences*, 231:98–112, 2013.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19, 2017.

Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

Gary Kacmarcik and Michael Gamon. Obfuscating document stylometry to preserve author anonymity. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 444–451. Association for Computational Linguistics, 2006.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. A dataset of peer reviews (peerread): Collection, insights and nlp applications. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, New Orleans, USA, June 2018. URL `https://arxiv.org/abs/1804.09635`.

Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. doi: 10.3115/v1/d14-1181.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94, 2011.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

Jaewoo Lee and Chris Clifton. How much is enough? choosing ε for differential privacy. In *International Conference on Information Security*, pages 325–340. Springer, 2011.

Hongmin Li, D Caragea, X Li, and Cornelia Caragea. Comparison of word embeddings and sentence encodings as generalized representations for crisis tweet classification tasks. *en. In: New Zealand*, page 13, 2018.

Ninghui Li, Wahbeh Qardaji, Dong Su, and Jianneng Cao. Privbasis: Frequent itemset mining with differential privacy. *Proceedings of the VLDB Endowment*, 5 (11):1340–1351, 2012.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*, pages 5103–5113, 2018.

Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 24–24. IEEE, 2006.

Andrew WE McDonald, Sadia Afroz, Aylin Caliskan, Ariel Stolerman, and Rachel Greenstadt. Use fewer instances of the letter i: Toward writing style anonymization. In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 299–318. Springer, 2012.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103. IEEE, 2007.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large datasets (how to break anonymity of the netflix prize dataset). *University of Texas at Austin*, 2008.

NhatHai Phan, Yue Wang, Xintao Wu, and Dejing Dou. Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

Joseph Rudman. The state of authorship attribution studies: Some problems and solutions. *Computers and the Humanities*, 31(4):351–365, 1997.

Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 81–98. ACM, 2011.

David Sanchez, Montserrat Batet, and Alexandre Viejo. Automatic general-purpose sanitization of textual documents. *IEEE Transactions on Information Forensics and Security*, 8(6):853–862, 2013.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 35–40, 2016.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841, 2017.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. A4NT: author attribute anonymity by adversarial training of neural machine translation. In *Proceedings of the 27th USENIX Conference on Security Symposium*, pages 1633–1650. USENIX Association, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Jordi Soria-Comas and Josep Domingo-Ferrert. Differential privacy via t-closeness in data publishing. In *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 27–35. IEEE, 2013.

Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and David Megías. Individual differential privacy: A utility-preserving formulation of differential privacy guarantees. *IEEE Transactions on Information Forensics and Security*, 12(6): 1418–1429, 2017.

Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology (JASIST)*, 60(3): 538–556, 2009.

Sandeep Subramanian, Guillaume Lample, Eric Michael Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. Multiple-attribute text style transfer. *arXiv preprint arXiv:1811.00552*, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

Veena Vasudevan and Ansamma John. A review on text sanitization. *International Journal of Computer Applications*, 95(25), 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016.

Benjamin Weggenmann and Florian Kerschbaum. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *Proceedings*

of the 41st ACM International Conference on Research & Development in Information Retrieval (SIGIR), pages 305–314. ACM, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P Xing, and Taylor Berg-Kirkpatrick. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3):378–393, 2006.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of*

the IEEE international conference on computer vision, pages 2223–2232, 2017.

Index

KEY TO ABBREVIATIONS