ETE-Anon: A Reinforcement Learning Based Framework for Textual Anonymization

Cameron Bale, Cassidy Klement, Burak Hamsioglu
Drexel University
CS 510 Class Project

Abstract

Much of the data that humans produce includes text data. While providing valuable insight in regards to data analysis, these data can pose a severe privacy risk to individuals, as well as violate data protection laws if not sufficiently anonymized. We propose a text anonymization framework that combines a sequence-to-sequence auto-encoder model with a reinforcement learning agent. This agent learns to produce synthetic text that preserves the sentiment and protects the privacy of the original text.

Introduction

Humans create a massive amount of text data. For example, there are approximately 500 million tweets sent daily[1], and Yelp has accumulated over 224 million online reviews[2]. Electronic health records (EHR) are commonly used to track patient care and information. These are valuable data sources for researchers and internet users, but the privacy risks these data pose to authors and data subjects are significant.

It has been known for several decades that stylometry, using statistical methods to analyze textual attributes, can be used to identify the author of a piece of text. A classic example is work by Mosteller and Wallace (1963), who utilized differences in word frequencies to determine the author of 12 of the Federalist papers. Today, individuals' privacy may be breached when statistical techniques are used to identify the authors of anonymous online posts, a process known as authorship attribution. Authorship attribution has even been used to identify the author of source code (Alsulami et al. 2017). This type of privacy breach is known as identity disclosure. The second type of privacy breach that can occur from text data is known as sensitive attribute disclosure (Beigi and Liu 2020). Attributes such as gender, age, race, and location may be inferred from or directly contained in a text. These attributes may further contribute to the threat of identity disclosure if they can be used to narrow down a set of potential authors. In addition to the properties of the text itself, the process used to create the text may be exploited to identify authors. Modern text recommendation systems can be exploited to observe the unique 'fingerprint' generated when a user is writing a piece of text (Sun, Schuster, and Shmatikov 2020). This fingerprint can be observed on a local device, then matched to anonymous online posts.

Of course, identifying attributes and authors of anonymous text can be used for non-malicious purposes. Businesses may wish to identify and contact the authors of negative posts so that their concerns can be addressed. These techniques may also assist with identifying plagiarized text or combating criminal activity[3]. However, text that is not anonymous poses a privacy risk to individuals and precludes data sharing in some fields. For example, under the United States Health Insurance Portability and Accountability Act (HIPAA), EHR must be de-identified by removing or replacing 18 categories of personal health information (PHI) before sharing. Since sharing EHR has significant benefits to medical research, recent research has focused on designing automatic de-identification systems that do not require de-identified text to be manually checked for correctness (Yogarajan, Pfahringer, and Mayo 2019).

Designing an end-to-end text anonymization system that will protect a given text against any privacy attack is a massive undertaking, and is outside the scope of this paper. We choose to focus on one portion of this task and address the problem of protecting the text from authorship attribution while maintaining the utility of the text for sentiment analysis. To tackle this problem, we propose an end-to-end anonymization system consisting of a reinforcement learning agent paired with an auto-encoder. This system takes a short text document as input and outputs an anyonymized version of the input text. The key component of our system is that the agent learns to perturb the context vector output from the encoder prior to passing the context vector to the decoder for text generation.

Our system is designed to protect the text (and its author) from authorship attribution and preserve the utility of the text for sentiment analysis. However, our training methodology and system design is flexible and can be trained to protect various privacy threats while preserving the utility of text for various applications. As will be described later, the system simply requires that the privacy threat and text utility can be explicitly modeled.

[1] https://www.internetlivestats.com/twitter-statistics/

[2] https://www.yelp-press.com/company/fast-facts/default.aspx

[3] https://www.fbi.gov/history/famous-cases/unabomber

Unfortunately, we were unable to achieve convergence in our estimate of the function used to guide the actions of our reinforcement learning agent. While the agent was successful in reducing the ability of an adversary to perform authorship attribution, the AUC for authorship attribution was still quite high. We consider the work in this paper to be a proof-of-concept and conclude this paper with several ideas that will be pursued to refine our training methodology and improve the ability of our framework to generate comprehensible anonymized text.

Related Work

Methodologies for protecting the privacy of textual data can be classified into three areas: protecting individual words, protecting text embeddings, and generating protected text. We provide brief reviews of these areas now.

Area One: Protecting Individual Words

Words that can be used to identify authors or infer sensitive attributes can be broken into three categories. The first is explicit identifiers (EID), which can be used to directly identify an individual. Examples include social security numbers, addresses, and patient identifiers. The second category is quasi-identifiers (QID), which cannot be used to directly identify individuals, but can be combined with external data to assist in identification. Examples include zip code, age, gender, and race. In addition to aiding in identifying individuals, some QIDs may fall under sensitive attributes, which is the third category. Sensitive attributes are any information an individual may not want to disclose publicly. Examples include income or whether an individual has a certain disease.

Previous approaches to anonymizing text, a process sometimes termed 'text sanitization', focused on replacing sensitive words (here, sensitive words refer to EIDs, QIDs, and sensitive attributes). Anandan et al. (2012) used hypernym trees to choose replacements for sensitive words that maintain a portion of the semantic quality of the text. This is known as generalization and these authors relied on the construct of *t*-plausibility: a document $\bar{d}$ with generalizations of sensitive words chosen from their corresponding hypernym trees is *t*-plausible if at least *t* texts could be generalized to $\bar{d}$. A similar technique was used by Feyisetan, Diethe, & Drake (2019) who anonymized text by choosing semantic generalizations for words in a manner that guaranteed $d_{\mathcal{X}}$-privacy, a generalization of differential privacy. EHR, Li & Qin (2017) proposed a value-enumeration method in which sensitive words were replaced with value-enumerated lists of similar terms to sanitize text.

One benefit to the methods in this category is that most of the original text will be unchanged. Hypernyms can be chosen such that the meaning behind the original text is largely preserved. However, the main drawback to these sensitive word-based approaches is that they may not prevent inferences that can be implicitly derived from the text. For example, usage rates of high frequency filler words such as *an*, *of*, and *upon* were strong discriminators on the basis of authorship of the 12 disputed federalist papers (Mosteller and Wallace 1963). Modern machine learning techniques could leverage this relationship to predict authorship even when sensitive words are generalized.

Area Two: Protecting Text Embeddings

The second area contains research into protecting text embeddings. Text is often converted into a vector or matrix representation for machine learning tasks. Mosallanezhad, Beigi, and Liu (2019) propose a novel Reinforcement Learning-based Text Anonymizer (RLTA) for generating privacy-preserving latent text representations. RLTA is comprised of four main components. The first is a bi-directional recurrent neural network (RNN) with gated recurrent units (GRUs) and a task-based attention layer (Luong, Pham, and Manning 2015). The RNN is trained to construct a latent representation of the input text, with the task-aware attention layer preferentially weighting the information that is most useful for the desired task, in this case, sentiment analysis. The second component is a reinforcement learning agent that is trained to perturb the context vector such that the utility for sentiment analysis is preserved while the ability to use the context vector to predict sensitive information (i.e., gender and location) about the original author is reduced. The third and fourth components aid in the training of the reinforcement learning agent: a private-attribute inference attacker model is trained to predict the gender and location of authors from the context vectors, and a utility model is trained to predict review sentiment from the context vector. The prediction probabilities outputted from these models are used to inform the agent's actions during training. The framework is trained and tested using real reviews from Trustpilot (Hovy, Johannsen, and Søgaard 2015). An important feature of RLTA is that the reward function used to train the reinforcement agent can preferentially weigh the privacy or utility of the latent representation. This is crucial in understanding and controlling the privacy-utility trade-off; however, RLTA only produces protected context vectors, which are not comprehensible to humans. For text such as online posts or medical text records to be truly anonymized, there is a need for methods that go beyond protecting embeddings and generate protected text. This is the focus of the third category of text anonymization methods.

Area Three: Generating Protected Text

The approaches in the third area generate privacy-preserving text. A recent effort by Krishna, Gupta, and Dupuy (2021) produced an auto-encoder, called ADePT, that is capable of generating differentially private text. The encoder and decoder portions of the auto-encoder were both unidirectional long short-term memory (LSTM) networks. The auto-encoder was initially trained without any privacy-preserving aspects using the ATIS (Dahl et al. 1994) and SNIPS (Coucke et al. 2018) datasets, both of which consist of short spoken language queries. The utility of this type of data is based on whether the intent of the query can be accurately classified. For example, [*Find me the I, Robot television show*] would have the intent label *SearchCreativeWork*[4]. The authors chose to measure the privacy of the lan-

[4]https://paperswithcode.com/dataset/snips

guage queries by measuring the performance of a membership inference attack (Shokri et al. 2017). The goal of this attack is to predict whether a given data sample was part of the data set used to train a machine learning model. In order to achieve differential privacy and protect against membership inference attacks, the authors tested the addition of both Gaussian and Laplacian noise to the context vector when it was outputted from the encoder, before passing it to the decoder. The intent classification accuracy on the private text was impressive - over 80% for all privacy budgets. However, ADePT was prone to producing corrupted outputs in which one word was repeated multiple times, and was trained on very short pieces of text. These limitations, combined with the strict privacy guarantees of differential privacy, means ADePT would likely have limited applications to longer pieces of text such as online reviews or EHR.

The framework we propose is contained in the third area of generating protected text. Our work builds directly on RLTA and ADePT by combining reinforcement learning and auto-encoding to generate privacy-preserving text. For reference, we include the table below to compare the aspects of our framework with those of RLTA and ADePT. We aim to combine the strengths of RLTA and ADePT and improve on the shortcomings of these approaches. First, we address the problem of full-text anonymization, not anonymizing latent text representations. We utilize a similar reinforcement learning approach to what is found in RLTA, with a slight modification to the reward function. Next, we utilize GRU cells in the RNNs that make up our auto-encoder in order to have good memory capacity with faster training than LSTM cells (Cho et al. 2014). For additional improvements in memory and the ability to handle longer sequences, we train our RNNs with three layers. It is important to note that Mosallanezhad, Beigi, and Liu (2019) use an auto-encoder to generate a privacy-preserving text embedding directly, and treat this as a baseline which RLTA is tested against. This auto-encoder was trained using a loss function that directly considered the utility and privacy of the embedding. However, we train an auto-encoder for input reconstruction and let the reinforcement learning agent take actions that improve the privacy of the generated text. We hypothesize that this will help maintain the utility of the anonymous text.

Approach

Consider a set $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots \mathbf{w}_N\}$ which contains N pieces of text. Each piece of text $\mathbf{w}_n = \{w_1, w_2, \dots\}$ is a sequence of words w_i. We propose a framework, denoted $\mathcal{F}$, which converts a text sequence $\mathbf{w}_n$ to an anonymous text sequence $\bar{\mathbf{w}}_n$. The conversion is performed to limit the ability of an adversary to infer the identity of the author of $\mathbf{w}_n$, while maintaining the usefulness of $\mathbf{w}_n$ for sentiment analysis. We denote an adversary's authorship attribution framework by $\mathcal{A}$ and an analyst's sentiment analysis framework by $\mathcal{S}$. The authorship attribution framework takes a text sequence as input and outputs a vector of probabilities of authorship for a set of P potential authors, $\mathcal{A} : \mathbf{w}_n \mapsto \mathbb{R}^P$. Similarly, the sentiment analysis framework takes a text sequence as input and outputs a vector of probabilities of the text having positive or negative sentiment, $\mathcal{S} : \mathbf{w}_n \mapsto \mathbb{R}^2$.

Data Preprocessing

We utilize the Yelp academic dataset[5], which contains over eight and a half million online reviews. Each review is associated with a specific user via a unique user identifier, a rating ranging from 1.0 to 5.0, and other author and review characteristics. We transform the review rating into a sentiment value by treating ratings of 4.0 or higher as positive sentiment and ratings below 4.0 as negative sentiment. Due to computational constraints, we reduce the initial size of the dataset by selecting the subset of reviews written by Yelp users that had submitted one thousand or more reviews[6]. This yields a new dataset containing 185,652 reviews.

The new set of reviews are further split to create training and testing sets for the auto-encoder, the reinforcement learning agent, and the authorship attribution and sentiment analysis models. First, we identify the 50 authors that had written the highest number of reviews contained in the new dataset. We find anywhere from 786 to 6073 reviews for each of the 50 authors, for a total of 62,030 reviews. Twenty percent (12,406) of these reviews are set aside for testing our framework, and we denote this testing set $\mathbf{W}^{test}$. The remaining 49,624 reviews from the top 50 authors are used to train the authorship attribution model, and we denote this training set $\mathbf{W}_{\mathcal{A}}^{train}$. We choose to provide a high amount of training data for each author in the authorship attribution model so that we can measure the ability of our framework to protect against authorship attribution in a worst-case scenario. In practice, an attacker may have much less training data for any given author.

We combine the 49,624 training reviews from the top 50 authors with the rest of the data, excluding the testing set. This gives a total of 173,246 reviews which we use to train our auto-encoder, reinforcement learning agent, and sentiment analysis and authorship attribution models. We denote this larger set of training reviews as $\mathbf{W}^{train}$. Note that $\mathbf{W}_{\mathcal{A}}^{train} \subset \mathbf{W}^{train}$. Due to computational constraints[7], we choose to analyze sequences consisting of 20 words from each review. We create these sequences by taking the first and last twenty words from each review in each training and testing dataset. This doubles the number of text sequences with which we can train and test our models, while keeping the training time for models at a manageable level.

Text Embeddings

Let $\mathbf{V} = \{w_1, w_2, \dots, w_V\}$ denote the set of all unique words found in $\mathbf{W}^{train}$. Prior to training the auto-encoder, each word $w_v \in \mathbf{V}$ is mapped to an embedding vector $\mathbf{u}_v \in \mathbb{R}^{300}$ using the GloVe (Pennington, Socher, and Manning 2014) Common Crawl 300-dimensional vectors for 840 billion tokens and 2.2 million unique words[8]. This results in a word embedding matrix $\mathbf{U} = [\mathbf{u}_1\, \mathbf{u}_2\, \dots\, \mathbf{u}_V]$. In cases where w_v is not found in the GloVe vocabulary, we create

[5]https://www.yelp.com/dataset

[6]The Yelp academic dataset does not necessarily contain all reviews written by a given user.

[7]Training our auto-encoder using full-length reviews was estimated to take several months to complete.

[8]https://nlp.stanford.edu/projects/glove/

Comparison of Text Anonymization Methods						
Framework	Year	Text Model	Privacy Model	Data	Attack Model	Utility Model
ETE-Anon	2021	Uni-directional GRU AE w/three layers	RL Agent	Yelp Academic Dataset	Authorship Attribution	Sentiment Analysis
ADePT	2021	Uni-directional LSTM AE	Differential Privacy	ATIS, SNIPS	Membership Inference	Intent Classification
RLTA	2019	Bi-directional GRU RNN w/attention layer	RL Agent	Trustpilot	Private Attribute Inference	Sentiment Analysis

an embedding $\mathbf{u}_v = [u_1\, u_2\, \ldots u_{300}]^T$ using 300 samples $u_j, j = 1, \ldots, 300$ where $u_j \sim N(0, 0.25)$.

Encoding and Decoding

The encoder and decoder portions of the auto-encoder model are trained as uni-directional recurrent neural networks with three layers consisting of GRUs. For a thorough exposition of the math behind RNNs with GRUs, see Chung et al. (2014). Consider a text sequence $\mathbf{w}_n = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_T\}$ in which the T words have been converted to their corresponding embedding vectors. The first layer of the encoder portion of the model takes each vector $\mathbf{u}_t$ as input in a sequential manner. For each input, the GRU in layer k calculates a hidden state $\mathbf{h}_{k,t+1}$ which is meant to capture dependencies across the entire sequence. For $t = 1$, the first layer of the encoder receives a randomized $\mathbf{h}_{1,1}$ and the vector $\mathbf{u}_1$. The hidden state $\mathbf{h}_{1,2}$ is calculated, passed as input to layer two, and maintained as the hidden state for layer one prior to receiving input $\mathbf{u}_2$. This process continues in a similar manner between layers two and three[9], and for $t = 1, \ldots, T$, at which point all embeddings $\mathbf{u}_t \in \mathbf{w}_n$ have been processed and a final combination of hidden states is outputted from the layers: in our case, a matrix of context vectors $\mathbf{H} = [\mathbf{h}_1\, \mathbf{h}_2\, \mathbf{h}_3]$ where $\mathbf{h}_k \in \mathbb{R}^{300}$, $k = 1, 2, 3$.

The hidden states in $\mathbf{H}$ are used as the initial hidden states of the layers for the decoder model. The first layer takes a start-of-string token as input for $t = 1$ and the third layer outputs predicted log probabilities over all words in $\mathbf{V}$. The highest log-probability is taken as the predicted word $\bar{w}_{t'}$. For $t' = 2, \ldots, T'$, the hidden state of layer k is passed as input into layer $k + 1$ for $k = 1, 2$, and the $\bar{w}_{t'}$ output from the third layer is passed as input into the first layer in time $t' + 1$. This process continues until the third layer predicts a special character to denote the end of the sequence. Note that the length of the predicted text sequence $\bar{\mathbf{w}}_n$ is not necessarily the same as the length of $\mathbf{w}_n$. During training, a sequence $\mathbf{w}_n \in \mathbf{W}^{train}$ is randomly selected and fed into the auto-encoder, which outputs the predicted sequence $\bar{\mathbf{w}}_n$ to minimize the negative log likelihood loss, $L(\mathbf{y}) = \sum_{t=1}^{T} -log(y_t)$ where y_t is the predicted log probability of the target word w_t. Examples of input text sequences $\mathbf{w}_n$ and output sequences $\bar{\mathbf{w}}_n$ can be seen in the appendix.

[9]Note that the output from the third layer of the encoder is ignored, while the hidden state calculations follow the same process.

Reinforcement Learning Agent

Next, we train a reinforcement learning agent to perturb the context vector $\mathbf{h}_1$ output from the first layer of the encoder at the end of the text sequence $\mathbf{w}_n$. We use s_t and $\tilde{\mathbf{h}}_t$ interchangeably to represent the context vector observed by the agent in time t. With some abuse of notation, we use t to index a time period for the agent, where the agent is allowed to perturb the context vector $\mathbf{h}_1$ a total of T times, once in each time $t = 1, \ldots, T$. In any time t, the agent observes the state s_t, and chooses an action a_t to maximize the total reward $R(\mathbf{s}) = \sum_{t=1}^{T} \gamma^{t-1} r_t(s_t)$, where $r_t(s_t)$ is the reward received in time t and γ is a discount applied to future rewards.

The agent is rewarded for preventing prediction of the correct author in the $\mathcal{A}$ framework and enabling prediction of the correct sentiment in the $\mathcal{S}$ framework based on the following function:

$$r_t(s_t) = \alpha C(s_t)_{\mathcal{S}} - (1 - \alpha)(C(s_t)_{\mathcal{A}} - \mathcal{E}(s_t)_{\mathcal{A}}),$$

where s_t is the state (new context vector $\tilde{\mathbf{h}}_t$) resulting from the action taken by the agent in time $t-1$. We use $C(s_t)_{\mathcal{S}}$ to denote the confidence of the sentiment analysis framework - the difference in predicted probabilities of the correct and incorrect sentiment:

$$C(s_t)_{\mathcal{S}} = P(l = i|s_t) - P(l = j|s_t)$$

where l is the sentiment label, i is the correct label (positive or negative), and j is the incorrect label. We use $C(s_t)_{\mathcal{A}}$ to denote the confidence of the authorship attribution framework - the difference in predicted probabilities of the true author and the non-true author with the highest predicted probability:

$$C(s_t)_{\mathcal{A}} = P(l = b|s_t) - max_{d \neq b} P(l = d|s_t)$$

where l is the author label, b is the correct author, and d is the incorrect author with the highest predicted probability of authorship. We use $\mathcal{E}(s_t)_{\mathcal{A}}$ to denote the scaled entropy of the authorship attribution prediction probability vector $\mathbf{a}$ that is output from the framework $\mathcal{A}$ based on state s_t:

$$\mathcal{E}(s_t)_{\mathcal{A}} = \frac{1}{log_2(P) * P} \sum_{p=1}^{P} -a_p log_2(a_p)$$

where a_p is the predicted probability of authorship for the pth author. This entropy is scaled by $log_2(P)$ to ensure a

maximum entropy of one. This reward function differs from the reward function utilized by Mosallanezhad, Beigi, and Liu (2019) in the subtraction of the scaled entropy term from the confidence of the $\mathcal{A}$ framework. We include this term so that the agent is incentivized to produce incorrect authorship attribution predictions and to create general uncertainty over the entire set of authors. $\mathcal{E}(s_t)_{\mathcal{A}} = 1$ when all authors have the same predicted probability of authorship. We scale this value by the number of potential authors so as to not outweigh the difference in predicted probabilities $C(s_t)_{\mathcal{A}}$. We believe this will encourage the agent to avoid states where two authors have very high probabilities of authorship, and the second highest is the correct author, which may constitute a privacy risk. Rather, we believe general uncertainty, not just an incorrect prediction, will protect author anonymity.

The reward function also utilizes an α value to allow us to preferentially weight the contribution of the sentiment prediction confidence $C(s_t)_{\mathcal{S}}$ or the authorship attribution confidence $C(s_t)_{\mathcal{A}}$ to the agent's reward $r_t(s_t)$. This type of parameter is valuable to privacy methodologies because it allows data stewards to select an optimal level of privacy verses utility for their use cases (Schneider et al. 2018); (Mosallanezhad, Beigi, and Liu 2019).

We use the prediction probability vectors from the sentiment analysis and authorship attribute frameworks as inputs to the reinforcement learning agent. It should be noted that the frameworks $\mathcal{A}$ and $\mathcal{S}$ make predictions based on the anonymized text sequence $\bar{\mathbf{w}}_n$ that is output from the decoder. The agent is trained as follows. A randomly sampled sequence $\mathbf{w}_n \in \mathbf{W}^{train}$ is fed through the encoder model. The agent observes the hidden state $\mathbf{h}_1$ output from the first layer of the encoder and selects an action to apply to this context vector. The agent has the option of changing any of the 300 values within the context vector to a number randomly drawn from one of three uniform distributions: $\mathcal{U}(-1.0, 0.9)$, $\mathcal{U}(-0.01, 0.01)$, or $\mathcal{U}(0.9, 1.0)$, resulting in a total of 900 actions available to the agent. We use a Deep Q-Network (DQN) to approximate the value function for the actions taken and rewards received by the agent.

In general, the agent observes some state s_t in time period t. An action a_t is selected and applied to s_t, producing a new state s_{t+1}, for which a reward r_{t+1} is received. The agent learns an action selection policy $\pi(s_t)$ to maximize the reward received by the terminal time period T. We can define a function $Q^* : S \times A \mapsto \mathbb{R}$ that maps the action taken in a given state to the total reward received, such that $\pi(s) = argmax_a Q^*(s, a)$. A Deep Q-Network is used to approximate the function $Q^*(s, a)$, by leveraging the Bellman equation:

$$Q(s_t, a_t) = r_{t+1} + argmax_a \gamma Q(s_{t+1}, a)$$

which can be re-arranged to form the temporal difference error[10] (Paszke nd):

$$\delta = Q(s_t, a_t) - (r_{t+1} + argmax_a \gamma Q(s_{t+1}, a))$$

[10]https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

The DQN is trained to minimize the huber loss (Huber 1964):

$$\mathcal{L}(\delta) = \frac{1}{|B|} \sum_{(s_t, a_t, s_{t+1}, r_{t+1}) \in B} L(\delta)$$

$$L(\delta) = \begin{cases} \frac{1}{2}\delta^2, & |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & otherwise \end{cases}$$

where B indicates a batch of state transitions randomly sampled from the agent's memory.

Sentiment Analysis and Authorship Attribution

We construct two linear support vector machine (SVM) models: one model exists in the framework $\mathcal{S}$ and predicts the sentiment of a given text sequence. The other exists in the framework $\mathcal{A}$ and predicts the author of the sequence.

The rest of the frameworks $\mathcal{S}$ and $\mathcal{A}$ consist of data processing steps applied to the text sequences. In both frameworks, the sequences are vectorized using a term frequency–inverse document frequency (TF-IDF) vectorizer, which computes the weighted frequency vectors for every review belonging in each class (words that occur more frequently are weighted more heavily). To reduce training and classification times, we use a linear SVM classification algorithm that minimizes the squared hinge loss, as opposed to the regular hinge loss, which converges significantly faster.

The SVM model in $\mathcal{S}$ is trained with $\mathbf{W}^{train}$ and performs binary classification, predicting whether a given text had positive (1) or negative (0) sentiment. The SVM model in $\mathcal{A}$ is trained with $\mathbf{W}_{\mathcal{A}}^{train}$. This model uses the one-vs-rest methodology, classifying a given text to the class (author) that has the highest probability.

Implementation

We train a DQN to accept an input vector of length 300 (context vector) with a hidden layer of 4200 nodes, and an output vector of length 900 - the estimated rewards from taking each of the 900 potential actions available to the agent. The agent chooses actions according to an epsilon-greedy algorithm, where a random action choice occurs with high probability in early stages of training (exploration) and the action with the highest expected return is chosen most often in the latter stages of training (exploitation). The agent is trained for 5000 episodes, where an episode consists of applying actions to the hidden state of a randomly selected sequence over $T = 25$ time periods. Each state, action, new state, and reward transition $(s_t, a_t, s_{t+1}, r_{t+1})$ is stored in the agent's replay memory, which holds up to 10000 of these transitions.

During training, one text sequence is randomly sampled, an action is applied, and the transition is stored in memory. Then a batch B of 32 previous transitions is sampled from the replay memory and used to update the DQN weights based on the huber loss. We calculate δ using a policy network $Q(s_t, a_t)$ and a target network $Q'(s_t, a_t)$ which is frozen to the initial policy network weights and is only updated every fifteen episodes. We compute $\delta = Q(s_t, a_t) - (r_{t+1} + argmax_a \gamma Q'(s_{t+1}, a))$ and use this to calculate the huber loss. The use of the target network is

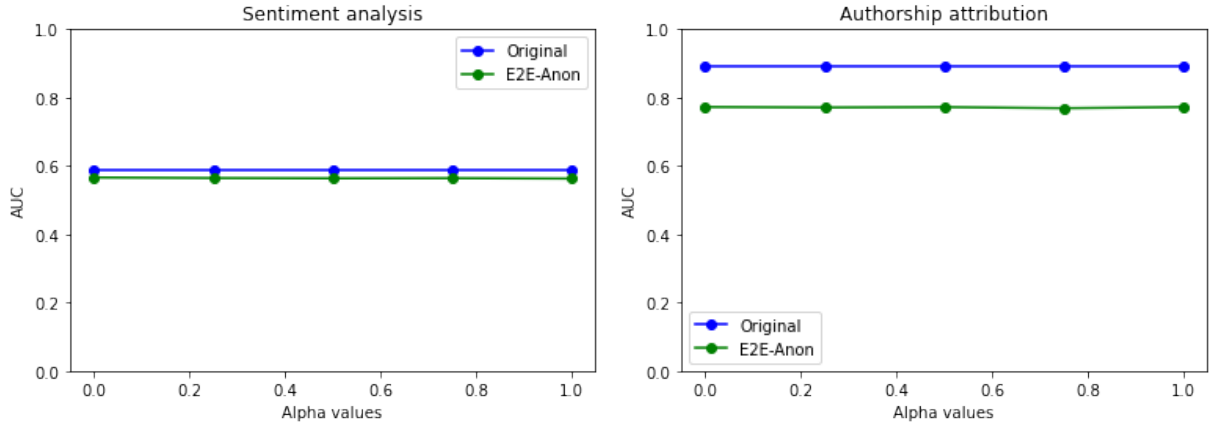

Figure 1: AUC scores for sentiment prediction & authorship attribution models for the original data set & varying values of α.

meant to stabilize the reward estimates and help the policy network $Q(s_t, a_t)$ converge to the true $Q^*(s, a)$.

We train a separate agent for each value of $\alpha \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ to compare the loss(gain) in utility(privacy) between different α values. The SVM models in $\mathcal{S}$ and $\mathcal{A}$ are implemented using scikit learn (Pedregosa et al. 2011) and our agent is implemented using PyTorch (Paszke et al. 2019).

Evaluation

To evaluate the five models trained with the chosen α values, we encode each sequence $\mathbf{w}_n \in \mathbf{W}^{test}$ using the trained encoder. For each encoded sample, we let the agent sequentially select $T = 25$ optimal actions for the given state. This is done for each of the five trained models and the final decoded samples $\bar{\mathbf{w}}_n \in \bar{\mathbf{W}}$ are returned.

As a baseline, we use the models in $\mathcal{S}$ and $\mathcal{A}$ to predict the sentiment and author of each of the original test sequences. We compute the area under the ROC curve (AUC) and the classification accuracies for these predictions. We repeat this process for the five sets of modified sequences yielded from the framework $\mathcal{F}$ based on the different values of α. These results are shown below.

Sentiment Prediction & Authorship Attribution Accuracies		
	Sentiment Analysis	Authorship Attribution
Original	68.46%	48.14%
$\alpha = 0.0$	62.13%	24.87%
$\alpha = 0.25$	62.43%	24.78%
$\alpha = 0.5$	62.34%	24.81%
$\alpha = 0.75$	62.48%	24.73%
$\alpha = 1.0$	62.38%	24.90%

The significant decrease in authorship attribution accuracy over all five models and the minor decrease in sentiment prediction accuracy indicate that all five models were successful in obfuscating the author of a given text and preserving the utility to some degree. However, the minimal variation in AUCs between the models indicates that our models are failing to converge, with the agents unable to learn a truly optimal action policy.

These results fall short of our expectations, and we believe this can be attributed to the computational constraints we faced throughout training and testing processes. We restricted the iterations for the training of several of the components, notably the auto-encoder and the DQN models. Since we limited the input text length to the models, as well, the available semantic content was significantly reduced.

Nonetheless, all five models are able to generate moderately comprehensible text as illustrated in the appendix. A considerable strength in our approach is the use of the scaled entropy term utilized in the reinforcement learning agent's reward function. By subtracting the entropy value from the authorship attribution confidence, we broaden the agent's goal with regards to authorship obfuscation to include general authorship uncertainty, potentially further protecting the author's anonymity, rather than focusing simply on the most probabilistic author being incorrect. The α value also utilized in the reward function, in the case that our models eventually converge, will prove to be a valuable method to allow for a higher contribution from the utility preservation or the authorship obfuscation. A higher α value would give more weight to the output from the sentiment prediction model when computing the reward, yielding a model that excels in preserving the utility with less attention given to protecting privacy. Conversely, a lower α value would yield a model that produces output with higher privacy.

Conclusion and Future Work

In summary, we propose an end-to-end anonymization system that utilizes an auto-encoder and a reinforcement learning agent perturbs the encoded version of a text sequence before the text is decoded. This system intends to protect the privacy of text while simultaneously preserving the utility. The concept of our framework could be applied for any type of utility case or privacy attack, given that these can be explicitly modeled and used to inform the actions of an agent.

Our results indicate that the agent is successful in its ability to reduce the likelihood of adversarial authorship attribution; however, the inability to converge illustrates that our

system has the opportunity for growth in several areas. First, we will continue to train the reinforcement learning agents for a significantly higher number of episodes, with more actions taken per episode. We may consider implementing an agent to perturb each hidden layer of our auto-encoder. Second, we will significantly expand the depth and training time of our auto-encoder so that it will be able to reconstruct much longer text sequences. There are several avenues of improvement that we have not explored (Hafner 2017) which we believe would significantly improve the comprehension of reconstructed text and the length of sequences our model could reconstruct.

We implemented relatively simple SVM models for sentiment analysis and authorship attribution. Once we obtain convergence for our agents, we plan to train them with more advanced sentiment analysis and authorship attribution models, for example, Koppel's algorithm for authorship attribution (Koppel, Schler, and Argamon 2011). Finally, we may consider alternative models for text generation and reconstruction. Recent deep learning advancements have produced transformer models (Lewis et al. 2019) that have immense natural language processing capabilities. We plan to explore this avenue as a potential alternative to an auto-encoder architecture.

References

Alsulami, B.; Dauber, E.; Harang, R.; Mancoridis, S.; and Greenstadt, R. 2017. Source code authorship attribution using long short-term memory based networks. *Lecture Notes in Computer Science* 10492 LNCS.

Anandan, B.; Clifton, C.; Jiang, W.; Murugesan, M.; Pastrana-Comacho, P.; and Si, L. 2012. T-plausibility: Generalizing words to desensitize text. *Trans. Data Privacy* 5(3):505–534.

Beigi, G., and Liu, H. 2020. *A Survey on Privacy in Social Media: Identification, Mitigation, and Applications*. Number 1 in 1. New York, NY, USA: Association for Computing Machinery.

Cho, K.; Merri ́enboer, B. V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; and et al., T. L. 2018. Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Dahl, D.; Bates, M.; Brown, M.; Fisher, W.; Hunicke-Smith, K.; and David Pallett, e. a. 1994. Expanding the scope of the atis task: The atis-3 corpus. *HLT Workshop*.

Feyisetan, O.; Diethe, T.; and Drake, T. 2019. Leveraging hierarchical representations for preserving privacy and utility in text. *CoRR* abs/1910.08917.

Hafner, D. 2017. Tips for training recurrent neural networks.

Hovy, D.; Johannsen, A.; and Søgaard, A. 2015. User review sites as a resource for large-scale sociolinguistic studies. *Proceedings of the 24th International Conference on World Wide Web*.

Huber, P. J. 1964. Robust estimation of a location parameter. *The Annals of Mathematical Statistics* 35(1):73–101.

Koppel, M.; Schler, J.; and Argamon, S. 2011. Authorship attribution in the wild. *Language Resources and Evaluation* 45(1):83–94.

Krishna, S.; Gupta, R.; and Dupuy, C. 2021. Adept: Auto-encoder based differentially private text transformation. *CoRR* abs/2102.01502.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, X.-B., and Qin, J. 2017. Anonymizing and sharing medical text records. *Information systems research* 28.

Luong, T.; Pham, H.; and Manning, C. 2015. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* 1412–1421.

Mosallanezhad, A.; Beigi, G.; and Liu, H. 2019. *Deep Reinforcement Learning-based Text Anonymization against Private-Attribute Inference*. Hong Kong, China: Association for Computational Linguistics.

Mosteller, F., and Wallace, D. L. 1963. Inference in an authorship problem. *Journal of the American Statistical Association* 58.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates, Inc.

Paszke, A. n.d. Reinforcement learning (dqn) tutorial. *PyTorch*.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1532 – 1543.

Schneider, M. J.; Jagpal, S.; Gupta, S.; Li, S.; and Yu, Y. 2018. A flexible method for protecting marketing data: An application to point-of-sale data. *Marketing Science* 37(1):153 – 171.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. *IEEE Symposium on Security and Privacy* 3–18.

Sun, Z.; Schuster, R.; and Shmatikov, V. 2020. De-anonymizing text by fingerprinting language generation. *CoRR* abs/2006.09615.

Yogarajan, V.; Pfahringer, B.; and Mayo, M. 2019. Automatic end-to-end de-identification: Is high accuracy the only metric? *CoRR* abs/1901.10583.

Appendix

Example Input & Output Text Sequences					
Input	$\alpha = 0.0$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1.0$
Though it would have been nice if they'd clean off the fallen leaves and branches more regularly.	Though it would have a reasonable then you would clean into the breakfast w and tons more once..	Though it would have a nice then you would clean into the breakfast w and were more happen..	Though it would have a nice if you would clean into the breakfast kept and bad way happen..	Though it would have a nice if you would clean into the breakfast walls and bad way else..	Though it would have a nice if you would clean into the breakfast kept and bad way happen..
Food is decent, service was surprisingly excellent and parking is a nightmare. I ordered the beef	Food is decent, service was certainly restaurant and parking is a notch. I ordered & and.	Food is decent, service only certainly food is parking is a winner. I ordered & and cheese.	Food is decent, service only certainly food is parking is a notch. I ordered food and cheese.	Food is decent, service only spacious and is this a notch. I ordered & and cheese.	Food is decent, service was surprisingly restaurant and parking is a winner. I ordered & and cheese.
What a great spot! Excellent service and the food is incredible very fresh with tons of flavor.	What a great spot! Friendly and the food is gorgeous very fresh with tons of taste..	What a great spot! Friendly and the food is gorgeous very fresh with tons of taste flavor.	What a great spot! Friendly and in food is incredible very fresh with tons of taste Good.	What a great spot! Friendly and the food is incredible very fresh with tons of taste friendly.	What a great spot! Friendly and the food is incredible very fresh with tons of taste flavor.
Again, my daughter and her friend declined to join us for lunch; and, wanted	Okay, my daughter and her brought decided to dine us for lunch; and, glad will not feel	Again, my daughter and her brought decided to dine us for lunch; and, glad will always feel	Once, my daughter and her her decided to dine us for lunch; and, glad will always feel	Once, my daughter and her boyfriend to come reservations for lunch; and, glad will always feel	Okay, my daughter and her her decided to dine us for lunch; and, glad will always feel