Measuring Unintended Memorization of Deep Generative Models on Email and SMS Datasets

Golshid Aflaki
11302120
Department of Decision Science
HEC Montréal
golshid.aflaki@hec.ca

Camille Duchesne
11242297
Department of Decision Science
HEC Montréal
camille.duchesne@hec.ca

Abstract

Insuring privacy for users is an important issue in machine learning. It is important to make sure that training data and user privacy is preserved. In this project, a privacy issue is studied for text generative models such as LSTM and GPT-2 on two datasets, NUS SMS and Enron email datasets, to find out if Deep Learning (DL) generative models, will memorize unintended information. In the training set, secrets represent personal information, and personal identification numbers such as SIN number which are there by mistake. The fact that this information can be memorized and leaked out by a DL model is a problem towards the privacy of the data and user, we are concerned to study if these two models are memorizing and if some cases such as the rareness of a secret, number of epochs in the training process, length of the secret text, and different representation techniques can affect the memorization pattern. A mitigation technique, Differential Privacy (DP) is studied to see if we can preserve privacy and prevent models from memorizing. We also looked at the relationship between accuracy and privacy (quantified by exposure) to see if they both chose the same model.

1 Introduction

Deep Learning models both discriminative and generative sequence models [1],[4] were shown to memorize unintentionally from the training data. This can be especially concerning for models which are trained on sensitive text data and even for data which is not necessarily sensitive in nature, it may contain some accidentally private information. For example, for a given model trained on email data, a person by inattention could have sent their social security number by email, now this highly sensitive data is in the dataset. This issue is extremely problematic in terms of privacy as generative models may generate sequences of sensitive information and thus leak the sensitive training data. If the model memorizes the SIN number, in the future when a user types the same sentence, the model can recommend that valid social security number. Obviously this is problematic, because if it generates a valid SIN number the model thus leaked highly sensitive information from the training data, the SIN can be used for any fraud reasons. The Secret Sharer [1] focused its work on recurrent neural network architectures. However, since the advent of the attention mechanism [16], transformer architectures have become the new norm and are widely used for their advantages in pre-training and their ability to measure more long-term dependencies.

It is important to note that this memorization is not a consequence of over-fitting [1], so the usual treatments such as early stopping and drop out are not going to be helpful to address this problem. Moreover, as [11] mentioned, “NLP training data are often mixed with sparse domain-dependent private information, and not all tokens need to be protected.” Thus, we chose not to “remove” potential sensitive tokens. Unintended memorization is an issue as it cannot be easily fixed with parameter tuning or simply censoring sensitive information.

We computed the unintentional memorization (exposure metric) of different deep and generative sequential models and tested different methods to minimize it using the Enron email dataset as well as the National University of Singapore SMS Corpus. These datasets were selected as they represent private conversations (emails and SMS) which could have contained sporadic private information. We will recreate this setting by artificially adding "canaries" or secrets to these corpora. We also seek to answer further research questions which are outlined in the following section.

1.1 Research Questions

Our main goal in this project is to study if the DL models such as LSTM and GPT-2 memorize unintended information. To do so, we explored the following research questions. The experiments design necessary to answer them are detailed in section 3, and result are shown in section 4.

1. How much unintended information do generative text models memorize? In this project, compare unintended memorization between an LSTM and GPT-2.
2. Does accuracy and exposure both choose the same model as the best model? The best model is often chosen only with respect to the accuracy, used to quantify its performance. Exposure metric could provide nuance to decide which model is the best in terms of preserving privacy. In this research question, we want to study if these metrics are positively correlated in the sense that the model chosen by accuracy is the same as the model chosen by exposure, or if there is a trade-off. `enumerate` environment.
3. How will secret rareness affect unintended memorization? If a secret only appears a few times in the training dataset, does the model memorize it immediately or does unintended memorization only happens for more repetitive secrets.
4. Can we mitigate the risk of memorizing information by making our algorithms differentially private?
5. Does the choice of feature representations influences memorization? How does it affect it?
6. Is there any relation between the length of a secret sentence and memorization?
7. Does the number of epochs have any influence on the model's memorization? We looked if more epochs may cause more memorization in DL models.

1.2 Our Contribution

We were concerned by the results of the Secret Sharer [1], we re-created their framework but incorporated other elements to contribute to the literature on unintended memorization in large language models.

- The Secret Sharer only used RNN-based models, but we are using a state of the art GPT-2 model in addition to LSTM to compare results in terms of memorization to see if more recent deep models also memorize unintended information since this model is more commonly used today.
- Implementing codes for GPT-2 itself and adapting the exposure metric to the model as it was not done in the Secret Sharer.
- We used the SMS dataset which was not used in [1], in addition to the Enron email dataset which was used in the Secret Sharer[1]. This dataset is similar to Email dataset in the way that they are both private conversational datasets in their structure. Additionally the cleaning, and preprocessing tasks was one of our contributions.
- Research questions 2,5, and 6 are other contributions to this body of work and are not answered in the Secret Sharer [1].
- Although a GitHub link was available in the paper, some sections of codes for this paper was available (not by the authors), there are some parts of the codes that we did by ourselves and was not available, notably the whole section on the GPT-2 model.

In the following sections, we will cover a literature review in section 2. In section 3, we review the design of our experiment. It contains information about datasets used, exposure metric, LSTM, and GPT-2 in addition to the DP-SGD as our mitigation technique. This section also covers our setup and experiment to answer each research question. In section 4 we have our results and in section 5 we explain our limitations and the possibilities of future work and improvements, we also studied different aspects of trustworthy ML related to our project and the societal impacts on our project in that section. Finally we concluded in section 6.

2 Literature Review

Carlini et al. [1] have found that deep neural nets can memorize unwanted information, notably that models can extract specific sequences of numbers from training data. They proposed the name unintended memorization and put forward the metric of exposure to quantify this problem in generative sequence models. They define unintended memorization as the capacity of certain trained neural networks to reveal out-of-distribution training data [1]. To measure unintended memorization, they inserted secrets or "canaries" into the training set and augmented the test set with all possible secrets to compute the exposure metric. This allowed them to compare the memorization in the training set versus random sequences of numbers which are represented by all the possible secrets in the test set. In this paper, we will utilize this framework to insert secrets in the training set and compute the exposure metric to compare different models and hyper-parameters. However, in the Secret Sharer paper, the secret length was of 9 random digits from 0 to 9 which lead to a very large search space for the test set. We adapted the secret length to 4 digits or 2 numbers up to 100 to reduce the search space. This paper used the Enron dataset, a very popular dataset in the field. We also used the sub-set of the Enron dataset and added a second, separate dataset, the NUS SMS dataset to compare the exposure on conversational data from formal emails and casual SMS data. Furthermore, in terms of their methodology, [1] reproduced their experiments and presented the average results, due to a lack of time and computing power, we could only run our tests once. They also tested the unintended memorization on LSTMs while we compared LSTM to a transformer model (GPT-2).

Building on this paper, Helali et al. [4] suggested a similar metric for discriminative sequence models named d-exposure. We will focus on measuring the exposure metric for text generative tasks, but it is interesting to see other ways in which this metric can be applied. Building on [1], Song Shmatikov [3] showed that text generation model memorizes more that numbers, and can even memorize words and sentences. These papers all used some form of recurrent neural network architecture to obtain their results. Our research is focused on the unintended memorization of numbers, we think that our work could be expanded in the future to compare the exposure of words and sentences in large language models. Paper [7] used canaries and transformers to measure unintended memorization, for automatic speech recognition task, we will do so on text generation task.

To mitigate the unintended memorization problem, [1] tried numerous techniques, notably regularization methods and differential privacy. They applied differentially-private stochastic gradient descent (DP-SGD) [8] and replaced the SGD optimizer by the RMSProp optimizer to obtain a higher accuracy. DP-SGD was implemented by clipping the per-example gradient to a max norm and adding Gaussian noise, they tested on multiple values of epsilon. Their results with DP was encouraging and showed that this DP-RMSProp does eliminate the unintended memorization for RNN models. We used the DP-SGD algorithm from the original paper [9]. From the mitigation techniques explored in [1], we only chose the ones which had showed promising results and applied DP-SGD to our LSTM. [10] Trained a RNN on a large public dataset and fine-tuned the model on a private dataset using DP-SGD. They showed that DP fine-tuning boosted the performance of their model compared to using an RNN trained with DP using only the private dataset. We did not have the computational resources to do this and pre-trained our LSTM on very large bodies of texts, we do find this method very compelling and for future research it would be interesting to compare the privacy difference between pre-training an LSTM on non-private data vs using DP which offers approximation of guarantees when using Gaussian noise. Because Transformer architectures generally scale up in parameters, [9] proposed a method to save memory by allowing clipping in DP-SGD to be executed without instantiating per-example gradients for all linear layer in the model. [11] Tested Selective-Differential-Privacy (SDP) [12] against DP-SGD for both Natural Language Understanding (NLU) and generating tasks. They worked with large language models such as GPT-2 [13] and Roberta [14] and fine-tune their models twice. They obtained better results with SDP than DP-SGD although SDP was found to be quite resource intensive and task specific. [6] Used pre-trained transformers (BERT and GPTs) on a variety of NLP tasks (including text generation) and developed a framework to efficiently fine-tune weights. They used DP to guarantee privacy with little to no impact on accuracy. [15]

Studied privacy issues around large-scale language model (BERT-large) and used DP-SGD during pre-training. Although we intended to compare the same DP-SGD algorithm used in [1] for our LSTM model and our GPT-2 model, we think these algorithms may make DP training for GPT-2 easier and could be explored in further research. This serves as our foundation to do a comparison study on the unintended memorization for RNN type architectures as well as large-scale language models and study potential mitigation techniques such as DP-SGD.

3 Design of experiment

3.1 Dataset

For our project, we used the Enron email dataset [17] featured in the Secret Sharer [1] as well as the National University of Singapore SMS Corpus (NUS) [18, 19]. The Enron dataset was collected and prepared by the CALO Project. It contains approximately 0.5M emails sent between 150 users. These messages were exchanged between employees at Enron between 1999 and 2003. The second dataset used is the NUS SMS Corpus, is composed of 55,835 short messages collected in 2015. The messages come from volunteers which knew their messages would be made publicly available. Private conversation data such as emails or SMS is naturally private but not always sensitive, indeed some conversations are mondain while others might be more delicate. Through the insertion of canaries, we aim at mimicking sparse sensitive information which might occur in private data (i.e. out of distribution data). As [1] mentioned, adding a secret (our canaries here) to a dataset that does not contain many secrets may obtain different results than working with a dataset which is very sensitive naturally, such as healthcare or financial data. This is a limitation of our project, due to its private nature, we could not get access publicly a dataset containing more sensitive data. It would be interesting for future work to measure unintended memorization of canaries in sensitive datasets in which these canaries would not be out of distribution.

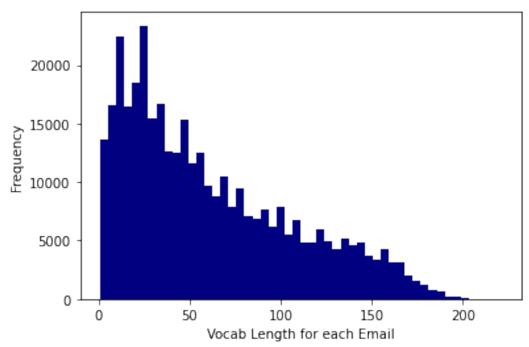
It is good to note that due to the lack of time and computational resources such as RAM and GPU, we failed to use all the information stored in both datasets so we used a portion of each dataset. For the SMS dataset out of approximately 55K messages, we used the first 30K messages. For the Email dataset after cleaning the data, and removing all unnecessary information about the time and date of sending, forwarding, or replying, emails with less than 70 characters were selected, we kept around 40K emails. We performed the cleaning process for both datasets, containing normalizing the text, removing punctuation, and white spaces. Then each dataset was separated into a training/testing/validation set. The first 20% of each dataset is set aside for the test set then from the remaining 80%, a portion of 20% was used for validation and the 80% left was for the training set.

Email data set: In table 1, we can see the statistics related to the word count in each email of the Enron dataset after data cleaning was performed. For the body of the email, the minimum number of words in an email is 1 and the maximum is 221, and the average length is 61 words. The distribution of the frequency of the different number of words in an email is shown in image 1a. We can see that the emails of length 25-30 are most frequent. The skewed distribution indicates that emails with a higher number of words are less frequent. Image 1c shows the word cloud of this dataset. The most frequent words are "Please", "Thank", as expected, we see words related to the office.

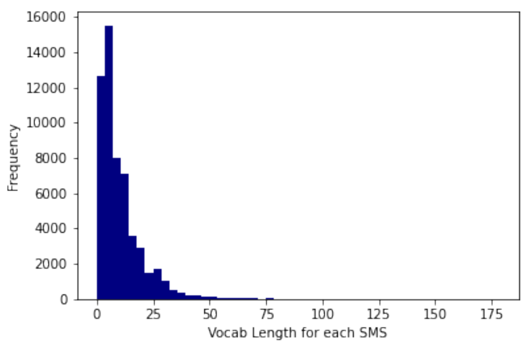
Table 1: Statistics of Email and SMS dataset, we can see the minimum, maximum and average number of words in each email/sms for each dataset.

	Min	Max	Average
Email	1	221	61
SMS	1	178	10

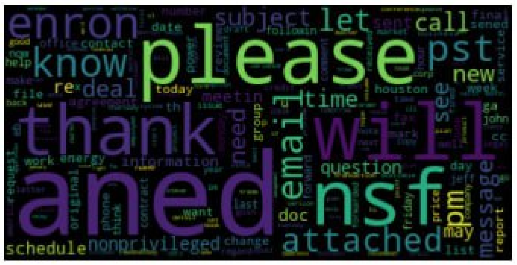
SMS data set: From table 1, we find that the minimum, maximum and average number of words in a text message is 1, 178, and 10, which is smaller than what we saw for the email dataset, which is expected. In figure 1b, we can see that text messages with around 10-12 words are the most frequent. Messages with more than 25 words are very low in frequency. In the SMS word cloud image 1d, the most frequently used words are "Okay" and "go", which is less formal than what we saw in the email dataset, which is again expected.



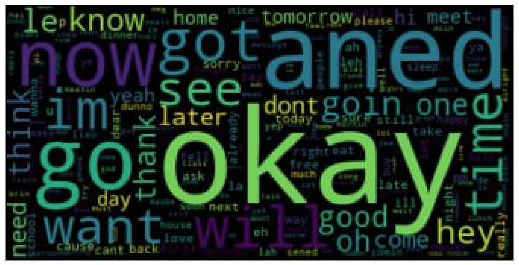
(a) Frequency of number of words appeared in each email in the Enron dataset



(b) Frequency of number of words that appeared in each message in the SMS dataset



(c) Word Cloud of the Enron dataset



(d) Word Cloud of the SMS dataset

Figure 1: Statistics of Enron dataset, image d and c in the left part and statistics of SMS data sets, image b and d, on the right side

3.2 Models, LSTM and GPT-2

Before the advent of attention mechanism and transformers, RNN based architectures were the backbone for NLP. Models with gated units such as LSTMs were commonly used for their "ease" in modeling long-term dependencies compared to non-gated RNNs. To compare our results with the Secret Sharer [1], and the GPT-2 model, we utilized a LSTM with 2 hidden layers with 100 units per layer. The model was implemented in Keras. As a baseline for the LSTM, because we couldn't beneficiate of the pre-training like for the GPT-2 model, we had to train the model on our datasets. However, to obtain the baseline for each dataset, we trained the model without inserting secrets in the training set, but inserting secrets in the test set and computing the exposure.

We were interested in measuring the privacy risk with these large language models compared to RNN architectures. Some companies such as OpenAI sell their APIs so other companies can benefit from pre-training and fine-tune the model using their data to have access to a performant, personalized language model. Released in 2019 by OpenAI, GPT-2 is a large language model whose architecture is akin to the decoder architecture of transformers. For this project, we used GPT-2 small to minimize our computational resources, it has 117 million parameters, it was trained on roughly 40 GB of internet text. We chose the GPT-2 model as a proxy for the newer GPT-3 model which was trained on even more data. Because the GPT-3 requires payment to access its API while it was free for GPT-2. To work with the GPT-2 model, we also utilized their pre-trained tokenizer ("GPT2") and augmented it with our datasets. We used the TFGPT2LMHeadModel as our pre-trained model which we fine-tuned. This is the TensorFlow pre-trained GPT-2 with a language model head on top. It can either be fine-tuned or used as is to directly generate text. This TFGPT2LMHeadModel pre-trained model without fine-tuning served as our baseline for our experiments. Without having seen any secrets, or any augmented data, the GPT-2 model evaluated the exposure of the test set with canaries.

3.3 Exposure Metric

In this project, the exposure metric, proposed in [1] is used to measure how much a DL model memorizes unnecessary out-of-distribution secrets. To use this exposure metric, some fake secrets are manually inserted in the dataset which are called *canaries* by [1]. Then the memorization of each model is computed based on these canaries and with the help of the exposure metric.

Canaries are random secrets. The canary is composed of two parts, a canary format, and a randomness part[1]. The canary format (S) is the fixed part of a canary, in this project we used "my permanent code is ** **". The random part of the canary is represented by *. They are numbers which were picked randomly (r). In our case, the random number completing the canary is a four-digit number. We create it by generating two, two-digit numbers between 0 and 99. An example of a complete canary (S[r]) is "my permanent code is 78 09". The randomness space R in this project is thus 100^2.

To actually measure unintended memorization, the exposure [1] of the inserted canaries is built based on log-perplexity which is defined below:

$$P_{X_\theta}(x_1, x_2, ..., x_n) = -log_2 Pr(x_1, ..., x_n | f_\theta)$$

$$= \sum_{i=1}^{n} (-log_2 Pr(x_i | f_\theta(x_1, x_2, ..., x_{i-1})))$$

In the equation above, the P_{X_θ} is the log-perplexity for the sequence of words which is $x_1, x_2, ..., x_n$. Log-perplexity is a likelihood-based metric for a sequence of words that shows how surprised a model would be in case of observing that specific sequence of words. As the value of log-perplexity gets bigger the model would be more and more surprised to observe this sequence. This metric is negatively correlated with the likelihood of observing a sequence of words (a sentence)[1].

The next step is the concept of the "rank of a canary", we rank all possible canaries based on log-perplexity. The first rank is dedicated to the canary with the smallest log-perplexity, or the predicted canary. All possible canaries represent all possible random numbers that can complete the canary format, this is the randomness space R. In our case, there are 10K canaries for which we computed the log-perplexity and then rank them. With the rank for the specific canary s[r], the exposure would be :

$$exposure_\theta(s[r]) = log_2|R| - log_2 rank_\theta(s[r])$$

[1].

In this equation, $rank_\theta(s[r])$ is the rank of the random canary S[r]. The maximum exposure happens when the rank of canary S[r] is one, or when the model predicted the secret. In our project, the maximum possible exposure is 13.2878 since the randomness space (R) is 100^2. A higher exposure means the model is memorizing the secrets more. We aim to minimize this value.

3.4 Differential Privacy (DP-SGD)

Privacy loss measures the difference in the performance of an algorithm given a data point is added or removed. Intuitively, if this difference is small, the privacy of that data point is preserved. To the contrary, if this difference is large, then the privacy of that user was violated, and the model leaked information about the training data. The idea behind ($\epsilon - \delta$) differential privacy is to bound this difference (privacy loss) while allowing some room to violate the bound (δ) to increase performance.

$$\mathbf{Pr}[A(D) \in S] \leq exp(\epsilon) \cdot \mathbf{Pr}[A(D') \in S] + \delta$$

Where A is the algorithm, D is a dataset, D' is the dataset slightly modified with either one extra data point or one less, privacy loss on a set of outcomes S, ϵ represents the upper-bound of the privacy loss or the privacy budget and δ represents a failure probability to relax the upper-bound. In our context, global differential privacy techniques are ideal for models which will be deployed publicly, such as GPT-2. By using Gaussian mechanism with ($\epsilon - \delta$) guarantees, we hope we can decrease the exposure problem while keeping a certain level of accuracy. As used in the Secret Sharer [1], the DP-SGD algorithm aims to limit the privacy loss per gradient update by modifying the optimization. The algorithm utilizes gradient clipping (which was already widely use with RNN to reduce the exploding gradient problem) to bound the gradient by the max L2 norm of C. It then adds noise to the clipping norm C before updating the gradient. It is thus possible to control the upper bound of our privacy loss, our privacy budget, by adding more or less noise. For the LSTM, we adapted the algorithm from [9], the original DP-SGD paper, and for the GPT-2, we wanted to used DP-SGD libraries but could not get results.

3.5 Set Up and Experiments

Our canary format S is "My permanent code is ** **", we randomize this canary format S by generating two, 2-digit numbers, they each can have values between 0 and 99. Then we define all possible canaries over the possible random numbers and then we pick one randomly to compute its exposure. Then random canary will be inserted into the training set, it can be inserted a different number of times depending on the research question we are working on and that will be specified for each research question later. After inserting secrets to the training data we will train our models on the augmented data. For each research question, the number of epochs and batch size is available in table 2.

Table 2: Number of Epochs and Batch Size for each research question

	#Epochs	Batch Size
Research Question 1	5	256
Research Question 2	5	256
Research Question 3	7	256
Research Question 4	5	256
Research Question 5	7	256
Research question 6	7	256
Research Question 7	5	256

For the **first research question** we want to study LSTM and GPT-2 to see if these models are memorizing unintended information. To do so, both datasets, SMS, and Email are used. For each dataset and each model, we add one specific canary, a different number of times to the dataset, and each time we compute the exposure. We insert a random canary 1, 2, 3, 4, 5, 10, 12, 15, 20, 30, 50, 100, and 200 times and train each model on the augmented dataset. Because the canary is random, for each number of insertions we redo the whole process 3 times for the LSTM and once for the GPT-2, and compute exposure each time. Finally, we compute the average exposure.

For the **second research question** we want to study if accuracy is enough and if the model that is picked based on accuracy is the model that leaks the least possible information. To do this both SMS and Email dataset are studied. From the first research question, we computed the accuracy as well as the exposure, we thus took the average accuracy over these iterations. We can thus compare the best model according to different criteria.

In the **third research question**, we want to study if the LSTM and GPT-2 are going to memorize even rare secrets, or if they need to observe the secret more frequently to memorize it. In order to address this question, only the SMS dataset is used. We insert a random secret different number of times in the dataset and compute the exposure each time to see if it augments. In this experiment, we insert a random canary 2, 5, 10, 20, 50, 100, and 150 times, then we train the models each time on the augmented dataset and compare exposures.

In the **fourth research question** we cover DP as a mitigation technique. We privatized the LSTM, with the DP-SGD optimizer and trained it on the SMS data. We expected to see a difference in exposure compared to the first research question. The hyper-parameters of the DP-SGD algorithm are $\epsilon = 1.5$, and $\delta = 0.001$. The setup follows the first research question in order to compare results.

Additionally, in **research question 5** we want to study if different feature representations affect how models memorize secrets. In this experiment, we use the SMS data and the LSTM model. In order to address this question we compare one-hot encoding with Glove, Fast Text, and Word2Vec which the last 3 are all prediction-based context-independent representation techniques. Then by inserting a random secret 10 times we train an LSTM model. Finally, we compute the exposure metric for each representation technique 3 times and we have the average exposure for each representation technique to compare. For the GPT-2 model, we tested with the roberta-base, bert and gpt2 embedding on the SMS dataset once per embedding.

In the **sixth research question**, we are trying to observe what will happen if the length of the secret changes and if there is any correlation between the length of a secret and the exposure. Before this research question for all other experiments a fixed canary format was used, which was "my permanent code is ** **" in which the prefix length was 4. In this research question, we are using different canary formats such as "my secret permanent code is ** **", and "my super secret permanent code is ** **" for which we have prefix lengths 5 and 6. In this experiment for each canary format, we choose a random canary and insert it 10 times in the train set, for each canary format we redo the process 3 times to have average exposure.The LSTM is trained on the augmented dataset and the exposure is computed for the SMS dataset, so that we can compare results.

Finally, in the **seventh research question**, we studied if there is any correlation between the number of epochs in the training process and memorization. To address this question we inserted a random secret 100 times in the training set, and we trained the LSTM based on the SMS dataset. We redid the whole process for each epoch 5 times and compute the average exposure. The number of epochs that are tried in this experiment is 2, 5, 7, 10, 20, and 30.

In the following section we will see the results of our 7 research questions.

4 Results

In this section results of the experiments designed in section 3.5 are available. For **Research Question 1** we can see results in table 3, for each model and each dataset there is an average exposure. Using LSTM the average exposure on the SMS dataset is 5.62 and for the email dataset it is 7.53 which is a relatively high exposure compared to the maximum exposure which is 13.28 so, we can conclude the LSTM model is memorizing secrets in both of the datasets. This is while the average exposure using the GPT-2 model on SMS dataset is 1.514 and 2.163 for the email dataset. The exposure is low and the model does not seem to memorize secrets in the training process. As a baseline, we considered a situation in which there are no secrets added to the train set, then the exposure is computed, and the exposure is very small.

Table 3: Average exposure and average accuracy for both SMS and Email datasets and LSTM and GPT-2 models and their baseline.

Models	LSTMS		GPT-2	
Data Sets	SMS	Email	SMS	Email
Average Exposure	5.6253	7.5380	1.5140	2.1625
Average Accuracy	4.82%	11.29%	52.95%	75.22%
No Secret Exposure(Baseline)	1.0115	0.7509	0.8953	0.6579

For the **Research Question 2** based on the results of the first research question, using the same experiment we have average accuracy in table 3 which shows the accuracy of the GPT-2 model is by far greater that the accuracy of the LSTM model. This says in terms of accuracy to perform a text-generating task, the GPT-2 outperforms the LSTM and we will pick GPT-2 over LSTM. Additionally, as it is discussed in the previous research question the GPT-2 model is memorizing less in comparison with the LSTM. So to address the research question we can say in this experiment, with these datasets and these mentioned setups the model that we pick based on accuracy is the same as what we pick based on privacy. It is good to note that this task is only comparing privacy and accuracy, there might be some other aspects like possible environmental damages or other expenses that lead us to pick LSTM with the less accuracy and tackle the privacy issue in another way.

From table 3, we can see that the results are consistent between the LSTM and GPT-2. Both show a higher exposure than the baseline, and report higher exposure and accuracy on the email dataset than on the SMS dataset. The GPT-2 model has a much lower exposure metric than the LSTM, this can be explained by the size of dataset-privacy-accuracy trade-off, with a larger dataset, it should yield higher accuracy, and when we used DP we would have to use less noise. According to our results, noise does not seem to be necessary as the size of the training dataset was so large for the GPT-2 model.

From iteration to iteration, the exposure metric remained highly volatile, so it is difficult to conclude if increasing the number of canary affects exposure for the GPT-2 model. We noted that contrarily to the exposure, the accuracy metric was not volatile, it was very stable, especially for the email dataset, it ranged between 74.30% to 76.31%. For the SMS dataset, the accuracy ranged from 65.75% to 76.14%, this could be due the quality of the English in the SMS. We can see that overall, the accuracy was better on the email dataset due to the English text being cleaner and with little to no slang.

Table 4: Highest 5 Ranks from the GPT-2 model on the email dataset with only 1 canary inserted

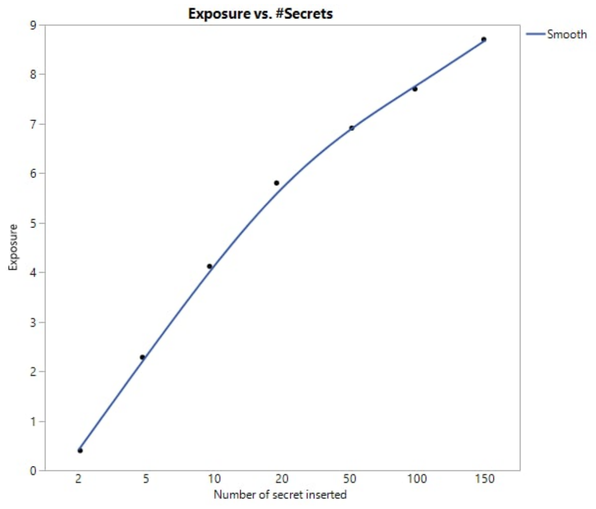
Rank	Secret #1	Secret #2	Actual Secret #1	Actual Secret #2
1	90	80	73	4
2	90	73	73	4
3	90	90	73	4
4	90	38	73	4
5	90	26	73	4

Table 4 shows the top ranks for the prediction of the canary by the model. These results come from the email dataset with only 1 canary inserted and obtained a 4.313 exposure which was quite high for the GPT-2 model. The canary was 73 04, and the top prediction would have been 90 80. Although this was one of the highest exposure the GPT-2 model provided, we can see that the top ranks don't necessarily correspond to the actual secret. This table was consistent across the GPT-2 model. Although volatile, even when looking at the model with the highest exposure score, it does not seem to memorize unintentionally. The model is not even close to predicting the secret canary. This goes against the results shown in the Secret Sharer [1], where their top ranks clearly showed that the top prediction was the actual canary, and the other highest ranks were variations of this canary.

Table 5: Average exposure for different numbers of secret insertions based on LSTM model trained on SMS

Number of Secrets	2	5	10	20	50	100	150
Exposure	0.398	2.283	4.120	5.804	6.913	7.703	8.703

Figure 2: Average exposure for different numbers of secret insertions based on LSTM model trained on SMS



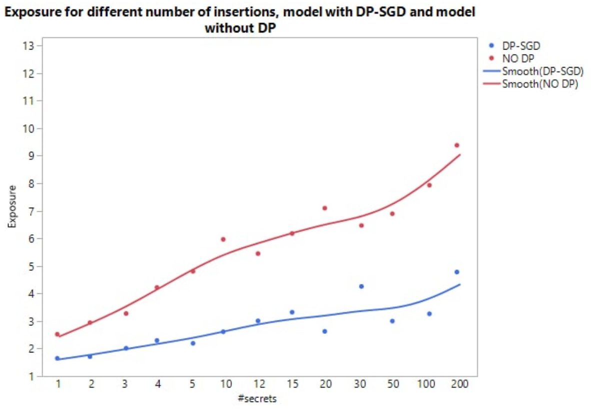
For **Research Question 3** based on results summarized in the table 5, using the SMS dataset and the LSTM model, we see that increasing the number of secrets added to the train set will increase the exposure. It is expected and understandable that as the model sees more of a secret, there is a bigger chance of memorizing because the model is more exposed to it. We can see when the model sees the secret rarely like 2 times in the train set the exposure is 0.398 which is very small compared to 13.28 which is the maximum exposure. In figure 2 we can see the semi-straight line that shows there is this linear relationship between the frequency of observing a secret in the train set and the relative exposure in the LSTM model.

For the GPT-2 model, we did not see a trend as we expected that the more canaries we inserted the larger the exposure would be. Indeed, inserting 1,100 and 150 gave us the highest exposure for both datasets while the insertion of 2,10,15,30 gave us exposure values close to 1. We conclude that we ought to perform more iterations of each insertion to obtain results which convey a trend.

Table 6: Average exposure and average accuracy of differentially private LSTM trained on SMS dataset. The noise is added by a Gaussian mechanism with $\delta = 0.001$ and $\epsilon = 1.5$.

Average Exposure	Average Accuracy
2.8254	3.1307

Figure 3: Exposure for different numbers of secret insertions in the train set while using LSTM with DP compared to LSTM without DP



In table 6 we show results for **Research Question 4**. We can see what is happening when we make our LSTM model differentially private trained on the SMS dataset. To use DP, we are using a DP-SGD optimizer and the noise is added based on the Gaussian mechanism with $\epsilon = 1.5$ and $\delta = 0.001$. We can compare results in table 6 with results in table 3 corresponding to results from research question 1. We used the same set-up but in figure 3 the LSTM is trained without DP. When we used LSTM without DP we had average exposure 5.62 for the SMS dataset. However when we use DP the average exposure is 2.82. In figure 3 the relatively lower exposure for all numbers of insertions, and the exposure hasn't exceeded 4 when we used DP. Plus, the slope is not as sharp as the LSTM without DP when we use DP-SGD. It means the exposure tends to stay constant even in a bigger number of insertions. For the LSTM without DP we can see the slope is greater with values near maximum exposure when 200 secrets were inserted. The LSTM with DP reaches the highest exposure around 4 when 200 secrets were inserted, which is comparable to the exposure of the LSTM without DP when only 3-4 secrets were inserted. For the accuracy while using DP, in table 6 we can see for LSTM with DP trained on SMS dataset, the average accuracy is 3.13 while it is 4.82 when the LSTM is without DP from research question 1 (see table 3). The changes in average accuracy may or may not be significant although we can see accuracy drops a bit by using DP, both exposure are extremely low. To confirm if this drop is significant we need to do more analysis and also have more iteration in both research questions to ensure these values are precise.

For the GPT-2 model, although we obtained results indicating there was no unintended memorization in **Research Question 1 2**, we still wanted to see the affect DP-SGD would have on the performance both in terms of exposure and accuracy. Unfortunately, we encountered problems both in PyTorch and in TensorFlow libraries, we ran into various issues with the DP-SGD algorithm from the Opacus and TensorFlow privacy libraries. We ran out of time for this project to code the DP-SGD algorithm for the GPT-2 model by ourselves.

Table 7: Effect of feature representation on average exposure (LSTM, trained on SMS dataset)

Feature Representation	Average Exposure
One-hot Encoding	4.4356
Glove	7.4554
FastText	4.9964
Word2Vec	4.4008

For **Research Question 5** we can see results in table 7 which shows exposure while we are using different feature representation techniques. In this project with all settings that we considered and for the LSTM model trained on the SMS data, inserting 10 times a specific canary, we can see the only representation that causes more memorization is the Glove representation. For other representations such as Word2Vec, FastText, and One-hot encoding there is no significant change in the exposure. For the GPT-2 model, we tested roberta-base, bert and GPT-2 pre-trained tokenizers, they did not make any difference on the exposure for both datasets.

Table 8: Effect of prefix length on average exposure (LSTM, trained on SMS dataset)

Secret Prefix	Prefix Length	Average Exposure
My permanent code is	4	4.4356
My secret permanent code is	5	5.5141
My super secret permanent code is	6	4.5909

Results of **Research Question 6** are stored in the table 8 in which exposure is related to different secret prefixes. We can see as the length of the secret prefix changed there is no significant change in the exposure. So this change in the canary format does not have much of an effect on the memorization while we are using LSTM trained on the SMS dataset.

Table 9: Effect of number of epochs on average exposure for LSTM trained on SMS dataset

Number of Epochs	2	5	7	10	20	30
Average Exposure	5.6397	6.6942	8.4383	8.0270	4.7897	4.8233

In the **Research Question 7** we are tracking the exposure while we change the number of epochs in the training process. As we can see the average exposure, is increasing until 10 epochs then it is decreasing. This is not what we expected. We believe it should be an always-increasing pattern by increasing the number of epochs. These results could be not precise because of the small number of iterations and the randomness of the canary. For the GPT-2 model, it converged often after 2 or 3 iterations. More precise results would be reached when we redo the experiment 100 times, due to the lack of time and computational resources is was not doable for this project. So these results could be more realistic by using more iterations.

5 Limitations and future works

5.1 Limitation

Our project has limitations, mainly the lack of time and computational resources, in order to improve this project with more resources, we can do several things.

1. Exposure metric is computed based on a random canary, so the exposure can be different if the canary which is picked and added to the training set was different. In that case, more precise exposure would be an average exposure on more than one random canary. In this project, we did our best in terms of time to compute average exposure but it was not for more than 5 iterations. More precise results could be achieved with more iterations.
2. The SMS and Email dataset are very large, to leverage their full size, we require better computational resources.To improve this project we could use the whole dataset. `enumerate` environment.

3. If there is still a need for using a portion of the data as we did, it would be better to have a better sampling strategy.
4. In the SMS dataset there are some Singaporean slang which impacted the models. Having a better strategy in the cleaning process may help to improve the precision of results in this project. And perhaps increase the accuracy of the models on this dataset.
5. In creating random secrets if we didn't have computational limitations, we could have greater a randomness space (which is now 10,000). In this case, the maximum exposure would be higher, and the differences of exposure for different insertion and in different research questions would be more visible.

5.2 Future works

As a future work to continue this research direction, we can suggest working on other state-of-art models to see if those are preserving privacy like how GPT-2 did. Also, prediction-based, context-dependent feature representation techniques can be studied, which were not studied in this project, to see if they change the memorization pattern in ML/deep models or not. Plus, in this project, we focused on generative deep models, in the future discriminative DL models can be studied as well in terms of memorization.

5.3 All aspects of Trustworthy ML

In the context of our experiment, it is difficult to evaluate if there was a disparity in treatment of the algorithm between groups since the protected attributes were not included in the data. It would of course be horrible if the model would memorize for a specific group more than others. In session 10 we saw that DP could both be aligned with Fairness goals and show individual fairness as a generalization of DP but could have contrasting goals as well where DP aggravates the disparate impact between groups. If the appropriate labelled data were accessible, it would be interesting to test the Fairness of our LSTM pre and post DP. The size of the protected group can play a role in the exacerbation of disparate impact, this is concerning since the paper of Bender & al. on the Dangers of Stochastic Parrots [20] shown that GPT-2 was trained on out-bound links from Reddit, it was shown in 2016 that 67% of Reddit users are men from the US, 64% of them in their twenties. For Wikipedia, its even worst, only 9-15% of the content would have been written by women. For protected attributes such as age and sex, it is very problematic since the GPT-2 model was mostly trained on data coming from young males from the US. In future work, it would be interesting to measure if the size of the protected group does have an affect on large language models and unintended memorization for text generation tasks.

A concern with robustness with the GPT models is that given how widely they are implemented if an attack were to be successful it could have unparalleled damage to a much larger pool of people. In terms of explainability, the methods need to be customized to text data. Some challenges include the interpretability of the word embedding and the model itself. These include the analysis of the model's understanding of the English language (i.e. via the Glue Benchmark), although these test sets mostly exist in English, in the context of our SMS dataset with English with slang from Singapore, these would not be currently available. We could also have attempted to provide explanation for the prediction. Looking at the rank of the canaries based on log-perplexity, the Secret Sharer paper [1] was able to provide some form of intuitive explanation as to why the model got a higher or lower exposure score. As we attempted this, I was not conclusive as we could see that the highest ranked were not close to the actual canary.

From the paper Concrete Problems in AI Safety [21], they outlined problems around machine learning systems and accidents which could lead to harmful behavior in reinforcement learning. They found that there was not sufficient testing for the potential errors and faults of the models before their deployment. We can extrapolate this to NLP, with the reach of these large language models, insufficiently test, could pose a serious issue for far more people than if only a local RNN were to fail.

Finally, briefly touching on the privacy aspect which was the focus of the paper, we were amazed that with the official release of GPT-2, there was no mention of privacy. The focus was solely on the performance regarding various language tasks. On one side, given that the main model was trained on text from the internet, if a person posted their social security number on a blog, the issue comes from the private data being publicly available rather that the GPT models could memorize it. However, for businesses or researchers which want to use the fine-tuned model, knowing it was not properly tested for privacy is very risky. We advise them to conduct the exposure test on their end before deploying the fine-tuned model to their customers/public.

5.4 Societal impacts

It is well known that training large models has a proportional large impact on the environment via the CO_2 emitted during their training. These large companies like OpenAI, google, etc. will throw massive amounts of computing power (energy) into training these ever-larger models to obtain small gains in accuracy. Indeed, one of the attractive features of the transformer architecture is their *stackability*, the ability to build these encode/decoder blocks on top of each other. Although they report high performance it comes as a double edge sword where these models are training on literally endless quantities of data. This is reflected in the sheer size of the GPT-3 model with 1.5B parameters, GPT-2 medium (345M parameters), GPT-2 large (774M parameters), etc. This level of access to resources is no longer accessible to students, most businesses, and large regions of the world. It centralizes in a way this privilege to access this tool. While the performance is rewarded, these closed API business model make it hard to trust the process in which it was trained and tested, especially for privacy concerns. With a lack of regulation in place, it is only up to these businesses to test and see if they are willing to take the reputational risk to launch an insufficiently tested model. As we write this, the new chat-GPT by OpenAI has taken traditional and social media by storm, its quality of text generation raises concerns around plagiarism in teachers, and potentially the loss of students' critical thinking. One can only hope that they sufficiently tested for privacy, robustness and fairness concerns. Other issues around large GPT models also include it being used to answer questions, fill in reviews, create text for a website, i.e populating the internet with their outputted text which is then used to train larger models. If these models were to have problematic behaviors, it creates a feedback loop.

6 Conclusion

This project was related to the privacy issue of text generation tasks, we looked at unintended memorization problem in the presence of out-of-distribution data in the trained dataset. We took the example of sending a piece of sensitive information like a SIN number in a text message to a family member. It is concerning that a text-generating model could suggest a SIN number to another user who types the exact prefix due to unintended memorization. That is a great concern, it can cause different forms of fraud. In this project, we studied the LSTM and GPT-2 models as text-generative models to see if they memorize sensitive out-of-distribution secrets. To this regard, SMS and Email dataset are used, from their nature they may contain sensitive information and we need to make sure models trained on these datasets can preserve the privacy of data and users. First, the average exposure for each of these models is computed, we showed that LSTM is memorizing sensitive information while GPT-2 is memorizes less. Additionally, according to the privacy and accuracy criteria, they both selected the GPT-2 model as the most performant model. To mitigate the memorization that is happening in the LSTM model we made our LSTM differentially private by using DP-SGD as an optimizer and the results show that DP helped to preserve privacy. Different cases were studied to see if they changed the amount of memorization. Examples of these cases are: the rareness of a secret, different representation techniques used in the training process, different numbers of epochs used in the training process, and different prefix lengths in the canary format. Results based on our experiment showed different prefix lengths don't have a significant effect on memorization. However, the rareness of a secret changes the amount of memorization. As the model sees more of a secret, it tends to memorize more (for the LSTM, we did not see this trend for the GPT-2 model). For feature representation techniques, our experiment shows there can be a difference between different feature representation techniques but we believe it should be studied more precisely with computing exposure of more secrets to having a more precise average exposure to decide if this change is significant or not. For the number of epochs, a more precise analysis is needed, but in our experiment, the results show till 10 epochs we have an increasing pattern in the exposure as the number of epochs increased, but after 10 epochs the average exposure drops for the LSTM model.

References

[1] N. Carlini, G. Brain, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, “The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks,” in 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 267–284. [Online]. Available: https://www.usenix.org/system/files/sec19-carlini.pdf

[2] O. D. Thakkar, S. Ramaswamy, R. Mathews, and F. Beaufays, “Understanding Unintended Memorization in Language Models Under Federated Learning,” ACLWeb, Jun. 01, 2021. [Online]. Available: https://aclanthology.org/2021.privatenlp-1.1/

[3] C. Song and V. Shmatikov, “Auditing Data Provenance in Text-Generation Models,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, Jul. 2019, pp. 196–206. doi: 10.1145/3292500.3330885

[4] M. Helali, T. Kleinbauer, and D. Klakow, “Assessing Unintended Memorization in Neural Discriminative Sequence Models,” in Text, Speech, and Dialogue, P. Petr Sojka, I. Kopeček, K. Pala, and A. Horák, Eds. Springer, 2020, pp. 265–272. doi: 10.1007/978-3-030-58323-$1_2$9

[5] W. Y. Wang, “‘Liar, Liar Pants on Fire’: A New Benchmark Dataset for Fake News Detection,” arXiv.org, 2017. [Online]. Available: https://arxiv.org/abs/1705.00648

[6] D. Yu et al., “Differentially Private Fine-tuning of Language Models,” arXiv:2110.06500 [cs, stat], Jul. 2022. [Online]. Available: https://arxiv.org/abs/2110.06500

[7] W. R. Huang, S. Chien, O. Thakkar, and R. Mathews, “Detecting Unintended Memorization in Language-Model-Fused ASR,” arXiv:2204.09606 [cs, eess], Jun. 2022. [Online]. Available: https://arxiv.org/abs/2204.09606

[8] M. Abadi et al., “Deep Learning with Differential Privacy,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS’16, pp. 308–318, 2016. doi: 10.1145/2976749.2978318

[9] X. Li, F. Tramèr, P. Liang, and T. Hashimoto, “Large Language Models Can Be Strong Differentially Private Learners,” arXiv:2110.05679 [cs], Oct. 2022. [Online]. Available: https://arxiv.org/abs/2110.05679

[10] G. Kerrigan, D. Slack, and J. Tuyls, “Differentially Private Language Models Benefit from Public Pretraining,” arXiv:2009.05886 [cs], Oct. 2020. [Online]. Available: https://arxiv.org/abs/2009.05886

[11] W. Shi, R. Shea, S. Chen, C. Zhang, R. Jia, and Z. Yu, “Just Fine-tune Twice: Selective Differential Privacy for Large Language Models,” arXiv:2204.07667 [cs], Oct. 2022. [Online]. Available: https://arxiv.org/abs/2204.07667

[12] W. Shi, A. Cui, E. Li, R. Jia, and Z. Yu, “Selective Differential Privacy for Language Modeling,” arXiv:2108.12944 [cs], Jul. 2022. [Online]. Available: https://arxiv.org/abs/2108.12944

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” OpenAI blog, vol. 1, no. 8, p. 9, 2019. [Online]. Available: https://life-extension.github.io/2020/05/27/GPT/language-models.pdf

[14] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” arXiv.org, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[15] R. Anil, B. Ghazi, V. Gupta, R. Kumar, and P. Manurangsi, “Large-Scale Differentially Private BERT,” arXiv:2108.01624 [cs], Aug. 2021. [Online]. Available: https://arxiv.org/abs/2108.01624

[16] A. Vaswani et al., “Attention Is All You Need,” in Advances in neural information processing systems, 2017. [Online]. Available: https://dl.acm.org/doi/10.5555/3295222.3295349

[17] B. Klimt and Y. Yang, “The enron corpus: a new dataset for email classification research,” ECML’04: Proceedings of the 15th European Conference on Machine Learning, 2004, pp. 217–226, [Online]. Available: https://dl.acm.org/doi/10.1007/978-3-540-30115-$8_2$2.

[18] T. Chen, K. Min-Yen, The National University of Singapore SMS Corpus, ScholarBank@NUS Repository, 2015. [Dataset]. Available: https://doi.org/10.25540/WVM0-4RNX

[19] T. Chen and M.-Y. Kan, “Creating a live, public short message service corpus: the NUS SMS corpus,” Language Resources and Evaluation, vol. 47, no. 2, pp. 299–355, 2013, doi: 10.1007/s10579-012-9197-9.

[20] E. Bender,T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 2021, pp. 610-623. Available: https://dl.acm.org/doi/abs/10.1145/3442188.3445922

[21] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," arXiv preprint , 2016. arXiv:1606.06565.