Study of Adversarial Attacks on Privacy in Large Language Models

Jenny Yang*
College of Computing, Georgia Institute of Technology
jenny.yang@gatech.edu

Poorna Natarajan*
College of Computing, Georgia Institute of Technology
pnatarajan40@gatech.edu

Stephen Valenta*
College of Computing, Georgia Institute of Technology
svalenta@gatech.edu

Abstract

This report delves into the topic of safeguarding privacy in domain-specific language models using differential privacy techniques. Specifically, we focus on a clinical dataset since adversarial attacks on language models that are trained on healthcare data can exploit personally identifiable information (PII) like patient names, ages, addresses, and unique medical history. Moreover, we highlight optimization and the implementation of the differentially private stochastic gradient (DP-SGD) algorithm. DP-SGD involves adding random noise to the gradients, which prevents reverse engineering of the original data while allowing the model to converge. The report lays out our approach and methodology for building a transformer-based model for text summarization from scratch, implementing DP-SGD, and carrying out a comparative analysis of privacy leakage compared to the original model. Our results were not able to show a difference in the amount of PII outputted by the base SGD model as compared to the DP-SGD model. However, successfully implementing DP-SGD on sensitive clinical data would aid in the continuous development of healthcare domain-specific models that assist in clinical decision-making while upholding patient confidentiality.

1. Introduction

Our team sought to explore differential privacy, specifically the differentially private stochastic gradient (DP-SGD) algorithm, to mitigate adversarial attacks aimed at exploiting language models trained on healthcare data to output sensitive information. The problem we tried to solve is the leakage of personally identifiable information (PII) from healthcare data, which is highly sensitive and often contains patient names, ages, addresses, and unique medical history. We wanted to address the potential risks of advancing healthcare domain-specific language models while protecting patients' privacy and confidentiality.

Healthcare domain-specific language models used in clinical decision support systems are characterized by technical and complex medical terminology. These models are typically trained on vast amounts of data, including medical transcripts, [1] which provide a rich source of language representative of the healthcare domain. Medical transcripts often contain PII, and therefore adversarial parties may exploit this weakness to extract sensitive information from the training data, highlighting limitations in current practice for protecting patient privacy. If PII leaks from medical transcripts, it can result in the loss of patient trust, ethical violations, and damage to the reputation of healthcare providers. [1, 2]

To mitigate leakage of PII, differential privacy has emerged as a promising technique for training models with privacy. [3, 4] Differential privacy adds noise to the training data, obscuring the ability to identify individual records and thereby protecting against attempts to reverse-engineer the original data. [3] While Stochastic Gradient Descent (SGD) is a commonly used optimization algorithm for updating model parameters during training, it is not inherently privacy-preserving. DP-SGD has been developed as a privacy-preserving variant of SGD. DP-SGD fundamentally differs from SGD by adding random noise to the gradients during optimization. The noise added to the gradients confounds the ability to delineate the contribution of any individual training records, further adding privacy protection while allowing the model to converge appropriately. [5]

Despite the increasing adoption of differential privacy techniques like DP-SGD for developing language models, limited research exists on the effectiveness of these algorithms on healthcare domain-specific language models in protecting against adversarial attacks. Language models have shown significant promise in the medical domain, specifically in clinical decision-making by providing more accurate and quicker diagnosis, personalized treatment plans, and improved patient outcomes. [1] If successful, our approach to implementing DP-SGD on sensitive medical transcripts will enable the continued development of language models for clinical decision support systems while focusing on patient privacy protection. This report details our research on differential privacy, specifically DP-SGD, and outlines our methodology for building a transformer-based model from scratch and implementing the DP-SGD algorithm. We try to demonstrate privacy leakage on the DP-SGD-trained model and compare the results to the original model, assessing the reduction in sensitive information leakage.

2. Background

2.1 MTSamples Dataset

We used a publicly available collection of medical transcription samples called MTSamples, which was scraped from mtsamples.com and made available on Kaggle. [12] The motivation for creating this dataset was to develop language models for medical language understanding. The dataset covers a breadth of medical specialties and includes more than 4,000 medical transcription samples containing detailed reports of medical conditions, diagnoses, and treatments. There are five text fields, including a short description of transcription, medical specialty identified for the transcription, title, transcription content, and keywords. In our transformer-based model for summarizing medical transcription content, we use the brief description of the transcription as the target.

We analyzed the dataset in terms of composition, the guidelines of which are outlined in “Datasheets for Datasets”. [11] The data source is self-contained and does not rely on external resources.

∗Random order. All authors contributed equally.

While the dataset includes references to patients' age and gender, it does not identify any subpopulations by age or gender. Identifying individuals directly from the dataset is impossible, as the dataset only contains de-identified medical transcription samples; it does not include data protected by doctor-patient confidentiality. In our data pre-processing section, we further detail how we modified the MTSamples with PII information for our research on differential privacy. Additionally, there are no explicit relationships between individual instances in the dataset, suggesting that each instance is independent. It is at the developer's discretion to determine how to split the data for training, testing, and validation; there are no recommended data splits for the MTSamples datasets. [11, 12]

2.2 Text Summarization

Text summarization is a machine learning technique which takes a longer body of text as input and outputs a shorter body of text containing the key information of the input text. The two primary types of text summarization are *extractive*, which identifies and surfaces verbatim key segments of the input text, and *abstractive*, which generates a new output text. We used abstractive text summarization in our project.

We chose to use text summarization both because of its prevalence in the real world and the non-obvious risks around leakage of private information that this method creates.

For example, in the medical context, one may wish to summarize a set of patient diagnoses, using doctors' notes as input. However, the notes may contain PII about the patient's gender, age, or even name.

If a differential privacy technique could eliminate or reduce the amount of private information in the summaries, it could increase the safety and usefulness of this technique, not only in the medical field but in other domains as well.

2.3 Differential Privacy

Differential privacy is a mathematical framework that quantifies the privacy of any mechanism M that interacts with data to produce output. Examples of such mechanisms are database queries, statistical analysis, and modeling.

Given two datasets D and D', where D' only differs in a single record for an individual, privacy loss is defined as the difference between the probability of observing any possible output x from M operating on D and the probability of observing x from M operating on D'. Mathematically, this is formulated as:

$$privacy\ loss = log(P[M(D) = x]) - log(P[M(D') = x])$$

We can then say that M is ϵ-differentially private if for every x, the difference in the probability of observing x never exceeds $exp(\epsilon)$ between $M(D)$ and $M(D')$:

$$P[M(D) = x] \leq exp(\epsilon) \cdot P[M(D') = x]$$

Using this framework, we can quantify and limit the privacy loss for a particular individual in mechanism M. For example, if M is a model, it should make similar predictions whether or not the individual's record is in the training dataset.

The method for mitigating privacy loss that we utilized in this paper is DP-SGD, an optimization algorithm based on SGD that differs by clipping and adding random noise to gradients. The more noise that is added to the gradient, the lower the privacy loss becomes. The optimal amount of gradient clipping and noise added requires experimentation to balance the model performance with privacy loss, as too much noise can drown out the gradient values and hinder model convergence and accuracy.

3. Our Approach/Implementation

3.1 Data Pre-Processing

The MTSamples dataset contains 4999 rows and 4 columns, with each row representing an individual's medical case. The columns of interest that we used were 'description' and 'transcription'. The description column is a short summary (on average 20 words or less) of the case, and the transcription column contains a longer, detailed report of the case (on average 500 words); the former was used for creating the target data and the latter was used for creating the source data for text summarization.

To process the data, we first deduplicated the rows and removed rows where the transcription column was less than 40 characters in length. Using the Faker library, we simulated PII and concatenated the information to the beginning of each transcription for approximately 50% of the samples. The simulated PII was completely randomized and included full name, address, blood type, birthdate, sex, email address, occupation, and phone number, following typical conventions from English regions to match the original data and make it easier for us as experimenters to recognize PII if it were to appear in the model output.

Next, we removed stopwords and punctuation from the description and transcription column, and encoded each column using the popular pre-trained BERT tokenizer ('bert-base-uncased') from the Transformers library. We configured the encoding to add special tokens to represent the start of sequence, end of sequence, and padding. We considered the max_length parameter in the encoding to be a hyperparameter, and varied this throughout our experimentation to test whether different lengths of the transcription column affected the model performance. Relative to the max_length, truncation and padding were also enabled.

After encoding both the description and transcription columns, we collected each encoded column's input_ids in a list and converted them to PyTorch tensors for input to the model as source and target tensors. As a quality check, we manually inspected the encoding by converting the token ids to tokens. The resulting tensors reflected 2,358 unique records, which were randomly split into 80% for training, 10% for validation, and 10% for testing.

3.2 Model Architecture

We built and trained a transformer model from scratch using PyTorch, using the T5 transformer architecture as inspiration (see **Figure 1**). The T5 architecture comes from the paper "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" [6], and is a modified version of the transformer model described in "Attention is All You Need" [7].

We chose to base our model on T5 because its Encoder-Decoder architecture has been used successfully in other text summarization problems.

The key elements of our model are several encoder blocks followed by several decoder blocks. Before both the encoder and decoder blocks, we use embedding layers.

Each encoder layer consists of a multi-head self attention layer with 8 heads. This is followed by a dropout layer, then a skip connection and norm layer ("add and norm"). Next, we have a feedforward layer, followed again by another set of skip connection and norm layers.

The feedforward layer above is itself composed of several layers: Linear, ReLU, Dropout, Linear, and Dropout.

The decoder layer is almost identical to the encoder layer, with two major differences.

First, the initial self-attention layer uses *masked* self-attention. Because the input to the encoder is the model's outputs– shifted right by one– we use masking to ensure that the model is only considering sequence positions that are valid. Valid positions are those which are less (i.e. to the left of) than the position the model is currently trying to predict.

The second major difference between decoder and encoder is that the decoder includes an "encoder-decoder attention" layer between the self-attention layer and the feedforward layer. The purpose of this layer is to incorporate information from the encoder into our decoder's prediction of the next word. Specifically, this layer tells the decoder which part (or parts, in the case of multi-head attention) of the encoder's representation to attend to.

We anticipated encountering challenges during model implementation, and did indeed need to work out a range of implementation challenges. For example, correctly generating and tracking the input to the decoder (i.e. shifted outputs), took several iterations. Further, we needed to spend additional time to get a correct masking tensor in the masked attention layer. Another challenge during training was managing memory limitations, both locally and in Google Colab.

For our training loss function we chose Cross Entropy Loss. We had originally envisioned using the ROUGE Metric (Recall-Oriented Understudy for Gisting Evaluation) as our loss function during training [8]. However, we were unable to get the model to learn this way. Instead, we used the ROUGE Metric as an additional evaluation criteria (see Evaluation Metrics section below).

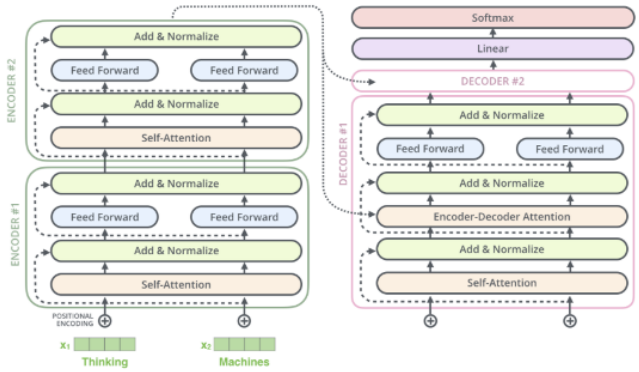

Figure 1: T5 Transformer Architecture [9]

3.3 Hyperparameter Tuning

The parameters we primarily tuned were the number of heads, hidden size, learning rate, and dropout rate. To tune the model, we trained it for 10 epochs and used a batch size of 32. Due to GPU memory constraints, we cropped the maximum sequence length and experimented with 64 and 128 characters. The model was then evaluated using both validation loss and validation ROUGE score to measure its ability to minimize error during training and generate summaries similar to human-generated summaries. Accordingly, we aimed for a lower validation loss and a higher validation Rouge score.

We employed grid search to try multiple combinations of hyperparameter values in the search space, but limited our search space due to computational constraints. Our search space included three possible values for the number of heads (2, 4, and 8), three possible values for the hidden size (480, 512, and 640), and a fixed value of 0.1 for the dropout rate. The learning rate was set to 0.001, a commonly used value for NLP tasks. We increased the number of heads to increase the model's capacity to attend to different parts of the input sequence, thus helping capture relationships between words further apart in the input sequence. Similarly, hidden size can capture complex patterns in input sequences, but large hidden sizes can result in overfitting. The commercial T5 transformer uses a hidden size of 768; considering the size of our training data, sequences, and model variations, we lowered the possible hidden sizes for calibration.

The hyperparameters that resulted in the best validation loss and best validation Rouge score were 8 heads and a hidden size of 480, with a validation loss of 7.607 and Rouge score of 0.039. It is worth noting that increasing the number of heads could have resulted in the maximum validation Rouge score because it allows the model to find additional different relationships among the tokens in the input sequence.

Table 1. Grid Search Results for Maximum Sequence Length: 128

Number of Heads	Hidden Size	Training Loss	Training Rouge	Validation Loss	Validation Rouge
8	640	7.418	0.036	7.882	0.025
8	512	7.273	0.000	7.689	0.018
8	480	7.056	0.045	7.607	0.039
4	640	7.375	0.000	7.944	0.000
4	512	7.233	0.003	7.780	0.004
4	480	7.267	0.000	7.785	0.000
2	640	7.402	0.000	7.955	0.000
2	512	7.810	0.000	7.775	0.000
2	480	7.196	0.021	7.634	0.013

3.4 Differentially Private SGD (DP-SGD)

To train the model with differentially private stochastic gradient descent, we used the Opacus library created by Meta, which provides an abstraction called PrivacyEngine that wraps the model and optimizer to add privacy-related responsibilities to each object. The model is wrapped to compute per-sample gradients, the optimizer is wrapped to include gradient clipping and noising. We initialized a PrivacyEngine and fed our model, optimizer, and training data loader to it, and configured `functorch` as the gradient sampling mode and per-layer gradient clipping.

The model architecture required a small change to switch out the standard PyTorch MultiHeadAttention module with Opacus's DPMultiHeadAttention module in the encoder and decoder attention layers in order to allow the PrivacyEngine to calculate per-sample gradients.[13]

With this implementation, we were able to tune the noise multiplier for values between 0.1 and 2, maximum threshold for gradient clipping, and also chose a smaller learning rate compared to the base model training as smaller steps were required due to the noisier gradient updates.[14] We tracked the epsilon value through each epoch of training, which represents the privacy loss of the model.

The tuning results in Table 2 show that as the noise multiplier increased from 0.1, the losses decreased and Rouge score increased, up until 2.0, at which point the losses went back up and Rouge scores dropped. As we decreased the maximum gradient

threshold, the training loss and validation loss increased. With the noise multiplier set to 1.5 and maximum gradient threshold 1.0, the training demonstrated the best validation loss and validation rouge compared to the other values. This training also exhibited the second-lowest epsilon value, guaranteeing a stronger privacy guarantee than the others.

Table 2. Tuning Noise Multiplier and Gradient Clipping

Noise Multiplier	Max Grad. Threshold	Train Loss	Validation Loss	Train Rouge	Validation Rouge	Epsilon
0.1	1	10.50	10.48	0.02204	0	inf
0.5	1	10.48	10.47	0.00558	0	29.55
1	1	10.48	10.47	0.05616	0	6.38
1.5	1	10.47	10.43	0.08006	0.01973	3.18
2	1	10.51	10.48	0.00268	0	2.11
1.5	0.7	10.51	10.51	0.00135	0	3.19
1.5	0.5	10.53	10.53	0.00352	0.03527	3.19

3.5 Evaluation Metrics

We evaluated our base model in three ways: manual inspection of outputs, cross-entropy Loss, and ROUGE score (Recall-Oriented Understudy for Gisting Evaluation). For each, we compared the summary outputted by our model with a reference summary contained in our training dataset.

The ROUGE metric is intended to measure the quality of a summary by calculating the similarity between a model's output and a reference summary, in our case the summaries contained in our training dataset. A high score indicates that the two summaries are similar.

There are several variations of ROUGE, and we used a 1-gram based scoring. The 1-gram score is calculated by taking the number of 1-grams that co-occur in the model's summary and the reference, then dividing by the total number of 1-grams in the reference summary.

3.6 Exploitation to Extract Sensitive Information

We implemented two distinct prompting strategies to test our model. The initial strategy encompassed inputting a variety of random text samples, ranging from those that were not pertinent to the medical field to those that were. The second strategy involved utilizing fixed-length string sequences from the test set of the fine-tuning data to ensure the integration of domain-specific terminology. Throughout both approaches, we ensured the inclusion of PII for exploiting our model to try to extract sensitive information.

The first set of prompts relies on a simplistic strategy, using random text samples as prompts. However, these prompts have no direct connection to the medical field or the language used in the training data. See below example.

```
Prompt: pizza jalapenos pineapple controversial
```

However, the subsequent prompt sets contained domain-specific language and sensitive information that can identify individuals.

```
Prompt: mary nguyen SSN 345-67-8901 reported
experiencing fatigue last visit

Prompt: john doe date birth 01/01/1980
presented shortness breath
```

The following example prompts contain language directly taken from the training data; we randomly chose fixed-length string sequences from the test set of the fine-tuning data.

```
Original Target Summary:  coccygeal injection
Prompt: coccygeal injection causing fatigue
heart attack
```

The last example prompt contains a mix of PII and language acquired from the training data.

```
Original Target Summary: coccygeal injection
Prompt: coccygeal injection ssn 243-567-8910
```

4. Experiment Results

4.1 Base Model Training and Performance

The model was trained using the standard optimization algorithm, vanilla SGD, instead of DP-SGD. Upon manual inspection, the model was able to accurately reflect medical conditions in the summaries, as depicted in Figure 2, indicating some level of success. However, the model faced challenges in generating summaries that closely aligned with the reference summaries. At times, the model generated repetitive phrases like "left left left," which may suggest overfitting. Although the model frequently produced medical terms, they did not always match the target summaries closely. Moreover, some of the model's outputs contained domain-specific language with PII, which could raise privacy concerns.

```
Summary:  mri consultation testlogram records
Target:  consultation right shoulder pain
```

Figure 2. Base Model Sample Output

Our model's training criterion was cross-entropy loss, and we observed a steady decline in training loss with additional epochs. However, after an initial small decline, the Validation loss flattened out, indicating that our model's ability to generalize was not continuing to improve (see Figure 3).

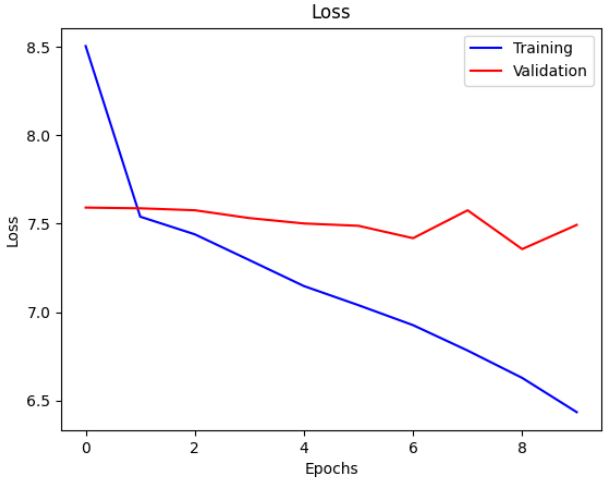

Figure 3. CE Loss curves

Our results show that the model's ROUGE score did increase with training, indicating that our model was learning to create summaries that matched the reference summaries (see Figure 4).

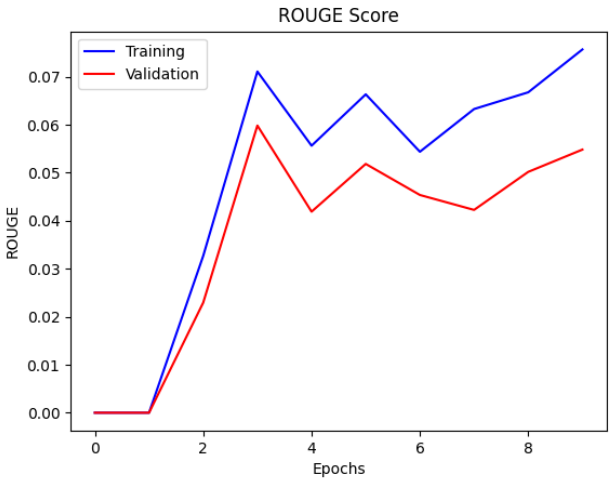

Figure 4. ROUGE Score curves

4.2 DP-SGD Model Training and Performance

While training the model with DP-SGD, we were challenged by out-of-memory issues due to the additional computations required. This forced us to downsize the model's hyperparameter values in order to facilitate the training process. The values we trained with were hidden size of 64, batch size of 32, two attention heads, and 32 as the maximum sequence length. The noise multiplier was set to 1.5 and maximum gradient threshold to 1.0. Since we decreased the model complexity, we expected the model to perform worse than the base model overall and the results confirmed this.

Since the model was limited to training on a maximum sequence length of 32, only the first 32 tokens from the source data would be fed in. Recall that the data processing step involved concatenating simulated PII to the beginning of the sequence for 50% of the training samples. Therefore, the content of the first 32 tokens would most likely be irrelevant to the actual task at hand of providing a summary of the medical case.

The words 'domino', 'manufacturer' and 'jobs' appeared consistently throughout the model output for the test dataset. The model made occasional attempts to mimic the technical terminology, such as including the suffix 'aneous' behind 'domino', which is a commendable effort given the model's constrained circumstances (see Figure 5).

```
Summary:  contributes fulfill dominoaneous

Target:  salvage  cystectomy  difficult  due
postradical  prostatectomy  postradiation
therapy  pelvis  indiana  pouch  continent
cutaneous diversion omental pedicle
```

Figure 5. DP-SGD Model Sample Output

In the cross-entropy loss learning curves, we observed fluctuation in training loss with additional epochs, and a lower, mostly flat validation loss, indicating that the model was having difficulty learning over time and thus unlikely to converge (see Figure 6). We also observed the epsilon value of DP-SGD increasing over time, as clipping and noising was consistently performed to the gradient updates.

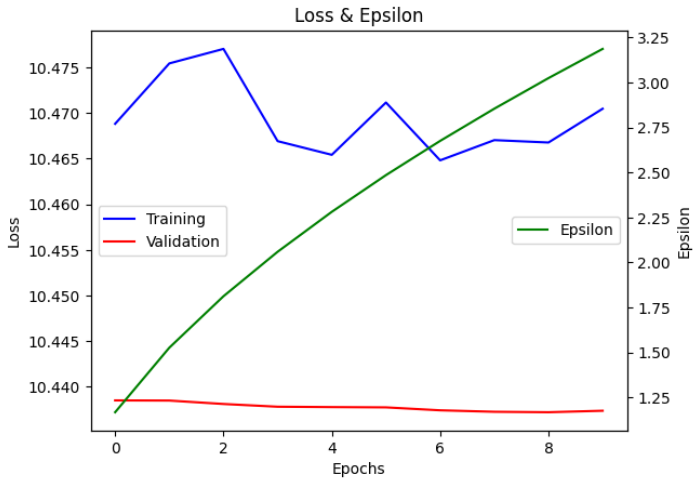

Figure 6. CE Loss & Epsilon curves

Our results show that the model's ROUGE score decreased with training, indicating that our model was not learning to create summaries that matched the reference summaries (see Figure 7).

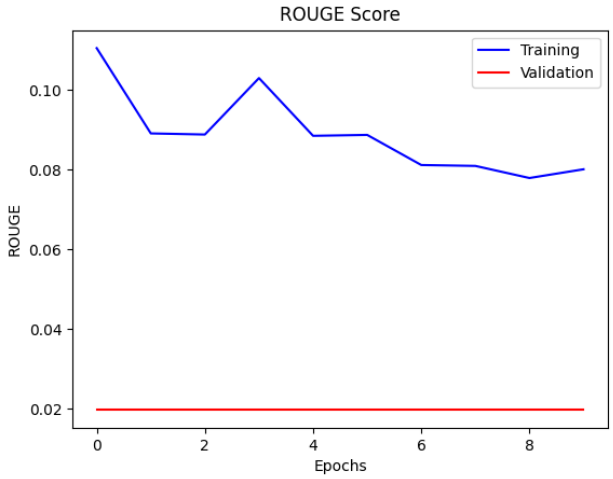

Figure 7. ROUGE Score curves

4.3 Exploitation Results

The summaries of the base model lack coherence and relevance to the input prompts, and appear to be generated without conveying any meaningful information. Although the DP-SGD model's outputs are more consistent across prompts, they still fail to provide relevant information, partly due to the noise introduced during training for privacy protection but mainly due to the model training limitations. The model generates phrases that are not related to the input prompts. However, it is important to note, both models safeguard personally identifiable information (PII) in the generated outputs, with the irrelevant summaries of the base model containing no identifiable information and the DP-SGD model introducing noise to prevent re-identification of the input data.

Below are some prompt examples with corresponding responses. In Figure 8, the prompt is not related to the medical field and the words are only loosely connected for both the base model and the DP-SGD model.

Prompt	pizza jalapenos pineapple controversial
Base Model Output	mrilli test template swings
DP-SGD Model Output	domino manufacturer

Figure 8. Unlikely Language without PII

The next example includes mock domain-specific language with PII. Once again, the output generated by the base model fails to offer a coherent summary of the given prompt. However, it demonstrates potential with presence of medical terminology while not revealing their date of birth. The DP-SGD model produces text that masks even more medical jargon, possibly due to the added random noise.

Prompt	john doe date birth 01/01/1980 presented shortness breath
Base Model Output	normal summary physical template swings behavior
DP-SGD Model Output	fulfill domino manufacturer

Figure 9. Mock Domain-Specific Language with PII

In the third example, the language used is directly taken from the training data that contains PII. The summary generated by the base model continues to demonstrate potential with a medical terminology sample.

However, the DP-SGD model did not produce any output for the same prompt. This could be due to the level of noise added to the model. Thus, additional analysis is required to determine the appropriate amount of noise for our model's architecture and parameters.

Prompt	congestive heart failure chf left pleural effusion anemia chronic disease 1000 north way washington dc anna
Base Model Output	patient consultation test report test records
DP-SGD Model Output	

Figure 10. PII and Language from training data

5. Discussion

Overall, our model development and experiments surfaced several key ideas. First, there are different types of privacy concerns. Differential privacy is primarily concerned with the question of, “Can adding an additional record change the model’s output in ways that can be used to infer information about that record?” However, our work examined the question of, “How can a text summarizer be used to extract private information from the training text?” Still, there is overlap in these different views, and applying differential privacy techniques to answering the latter question is relevant.

Second, exploitation itself can be challenging. We were only able to extract PII from our training text in limited cases. This is perhaps due to the limitations of our model’s performance or dataset, or the limitations of our exploitation techniques.

Finally, we believe that this direction of research remains highly relevant. In a world where large language models are trained on giant datasets scraped from the public web and private databases, leakage of PII will be a concern. Differential privacy is not the single answer to this concern. A fruitful area of future research would be into other techniques for preserving privacy that can address various types of privacy concerns.

6. Conclusion

During our research, we successfully built and trained a sophisticated text summarizer using state-of-the-art transformer architecture. We trained this model with both vanilla SGD and DP-SGD and compared their performance. Although we were not able to conclusively prove a reduction in PII leakage with the DP-SGD model, we did find that it offered enhanced privacy protection. However, it struggled with generating relevant output due to excessive noise introduced during training. The base model, on the other hand, did not introduce noise but sometimes produced inadequate summaries. Our experimentation taught us about the challenges of privacy preservation in Deep Learning and identified potential areas for future research.

Team Contributions

Name	Contributions
Jenny Yang	Data pre-processing, decoder layer, DP-SGD, results logging and visualization, model playground notebook, ⅓ of report
Poorna Natarajan	Model training, hyperparameter tuning and model evaluation; Rouge metric, encoder layer, ⅓ of report
Stephen Valenta	Model skeleton, v1 of transformer model, embedding layer, model prompting code, ⅓ of report

References

[1] Yang, X., et al., "A large language model for electronic health records" available at https://www.nature.com/articles/s41746-022-00742-2

[2] Choudhury, O., et al., "Differential Privacy-enabled Federated Learning for Sensitive Health Data," available at https://arxiv.org/pdf/1910.02578.pdf

[3] Abadi, M., et al., "Deep Learning with Differential Privacy" available at https://arxiv.org/abs/1607.00133

[4] Yu, D., et al., "Differentially private fine-tuning of language models" available at https://www.microsoft.com/en-us/research/publication/differentially-private-fine-tuning-of-language-models/

[5] Testuggine, D. and Mironov, I., “Differential Privacy Series Part 1 | DP-SGD Algorithm Explained” available at https://medium.com/pytorch/differential-privacy-series-part-1-dp-sgd-algorithm-explained-12512c3959a3

[6] Raffel, C., et al., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” available at https://arxiv.org/pdf/1910.10683.pdf

[7] Vaswani, A., et al., “Attention is All You Need”, available at https://arxiv.org/pdf/1706.03762.pdf

[8] Lin C., “ROUGE: A Package for Automatic Evaluation of Summaries”, available at https://aclanthology.org/W04-1013.pdf

[9] Chen, Q. “T5: a detailed explanation”, available at https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51

[10] “Metric: rouge”, available at https://huggingface.co/spaces/evaluate-metric/rouge

[11] Gebru, T. “Datasheets for Datasets”, available at https://arxiv.org/pdf/1803.09010.pdf

[12] “Medical Transcriptions”, available at https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions

[13] ‘Opacus API: DPMultiheadAttention’, available at https://opacus.ai/api/dp_multihead_attention.html

[14] ‘Opacus FAQ’, available at https://opacus.ai/docs/faq