Privacy Adhering Machine *Un-learning* in NLP

Vinayshekhar Bannihatti Kumar, Rashmi Gangadharaiah, Dan Roth

AWS AI Labs

{vinayshk,rgangad,drot}@amazon.com

Abstract

Regulations introduced General Data Protection Regulation (GDPR) in the EU or California Consumer Privacy Act (CCPA) in the US have included provisions on the *right to be forgotten* that mandates industry applications to remove data related to an individual from their systems. In several real world industry applications that use Machine Learning to build models on user data, such mandates require significant effort both in terms of data cleansing as well as model retraining while ensuring the models do not deteriorate in prediction quality due to removal of data. As a result, continuous removal of data and model retraining steps do not scale if these applications receive such requests at a very high frequency. Recently, a few researchers proposed the idea of *Machine Unlearning* to tackle this challenge. Despite the significant importance of this task, the area of Machine Unlearning is under-explored in Natural Language Processing (NLP) tasks. In this paper, we explore the Unlearning framework on various GLUE tasks (Wang et al., 2018), such as, QQP, SST and MNLI. We propose computationally efficient approaches (SISA-FC and SISA-A) to perform *guaranteed* Unlearning that provides significant reduction in terms of both memory (90-95%), time (100x) and space consumption (99%) in comparison to the baselines while having minimal impact on model performance.[1].

1 Introduction

The use of user generated content in building ML-based solutions in many industry applications is quite common. As new data is acquired, these ML models undergo constant updates to further improve the performance. Such data can contain sensitive information such as names, account ids, email addresses and so on. Typically, user generated content goes through several redaction systems to remove such sensitive information before being used for retraining purposes. Industry applications also adopt rigorous approval processes with opt-in or opt-out capabilities to ensure that user generated content is used in a safe manner while raising awareness among users on how their generated content will be used.

More recently, the GDPR and CCPA provisioned the right to be forgotten, introducing new regulations that mandates industry applications to support deletion of user generated content when requested by users. Although privacy concerns have gained significant importance now, the scale at which these models are built make it extremely expensive and time-consuming to remove data efficiently while providing *complete removal guarantees*.

Unlearning assures the users that their data has been completely removed when they prefer to have their data erased (Bourtoule et al., 2019). This task is different from Differential Privacy that focuses on models that ensure that the users' information in the training data cannot be inferred and provides guarantees on the effect of an individual examples. Such a task is challenging as removal of data points can have significant deterioration on the performance of the models *aka catastrophic unlearning* (Du et al., 2019; Golatkar et al., 2019; Nguyen et al., 2020, 2022a).

Approaches that guarantee complete removal of users' data points have recently gained momentum (Bourtoule et al., 2019; Nguyen et al., 2022b) but are under explored on NLP tasks. Motivated by SISA (**S**harded, **I**solated, **S**liced and **A**ggregated) training (Bourtoule et al., 2019) applied to CV datasets, we explore an approach that trains multiple ML models in isolation on disjoint shards and its slices. Model checkpoints are saved after training each slice in a shard. When a request for deletion is received, the corresponding datapoint is deleted from its slice and the model checkpoint upto the datapoint is used to further retrain

---

[1]https://github.com/awslabs/privacy-adhering-machine-unlearning-nlp

the model. Although the SISA framework can be applied to any learning algorithm, it is still impractical in NLP settings as storing model checkpoints that include large language models (such as, BERT (Devlin et al., 2018a)) is both space and time consuming. We propose extensions to this SISA framework (1) (**SISA-FC**) requires storing only task-specific layers (2) (**SISA-A**) uses Adapters (Houlsby et al., 2019) that requires storing only the adapter weights. These extensions prevent storing the entire model checkpoints, hence reducing time, memory and space footprints. We further improve upon the approaches by creating shards such that the least number of slices are affected when requests for removal are made, further reducing the time required to retrain the models. To the best of our knowledge this paper makes the first attempt to explore the task of Unlearning on various NLP tasks. The contributions of this paper are summarized as follows:

- We explore the task of Unlearning on various NLP tasks (GLUE tasks such as QQP, SST and MNLI) in both full data as well as few shot settings.

- We explore SISA-FC and SISA-A for NLP-based models that do not require storing large model checkpoints, thereby significantly reducing space, memory and time

- We propose novel ways to partition the data, thereby reducing the number of slices affected and the retraining time while ensuring minimal degradation in the overall model quality.

The paper is organized as follows. Section 2 briefly describes some of the related work in the area of Unlearning. Section 3 describes the NLP datasets used to evaluate the approaches proposed in this paper. Section 4 explains the proposed approaches (SISA-FC and SISA-A) followed by results in Section 5. We finally conclude and provide future extensions to this work in Section 6.

2 Related Work

Privacy preserving ML provides guarantees on bounds ensuring the contribution from the data points is as low as possible (Sarwate et al., 2009; Abadi et al., 2016; Jang et al., 2022; Yu et al., 2021; Li et al., 2021). Machine *Unlearning* on the other hand focuses on *complete* removal of training examples, ensuring that there is *zero* contribution of

the training sample to the model's learned weights. As a result, Unlearning assures the users that their data has been completely removed when they prefer to have their data erased (Bourtoule et al., 2019). A simple approach to forget training samples would be to re-train the models after removal of the examples. Such an approach is not scalable and computationally expensive when systems receive removal requests at high frequencies. Another challenge this brings is in the deterioration of the performance of the models.

Machine Unlearning has gained significant attention in industry applications as a means to allow users to completely delete their data from ML models (Tahiliani et al., 2021; Nguyen et al., 2022a; Baumhauer et al., 2020). Regulations introduced by regulatory bodies apply to all forms of user generated content. However, most of the recent approaches tackle this problem of Unlearning in Computer Vision (CV) settings (Mehta et al., 2022; Golatkar et al., 2019; Bourtoule et al., 2019). It is imperative to explore unlearning strategies on textual data that can contain user sensitive information or personally identifiable information (PII).

Our work is inspired by SISA training (Bourtoule et al., 2019) which was originally applied to CV datasets. The framework provides a strategic way to limit the influence of a data point in the training procedure. The approach trains models in isolation on disjoint shards created by partitioning the training data. When a request for removal is made, only the affected model is retrained. Shards are further broken down into slices to decrease the time required to unlearn. During inference, they use ensemble strategies to aggregate predictions of individual models. We extend this framework for NLP models. Our paper also proposes approaches to partition the data such that the least number of shards/slices are affected, thereby further reducing the re-training time.

3 Datasets

Glue (Wang et al., 2018) tasks are divided into three categories namely , "SINGLE-SENTENCE TASKS", " SIMILARITY AND PARAPHRASE TASKS" , " INFERENCE TASKS". We pick "SST-2" dataset from the first, "QQP" dataset from the second and "MNLI" dataset from the third respectively. In order to test the generalizability of our approach in terms of performance, time taken to retrain models and the memory they
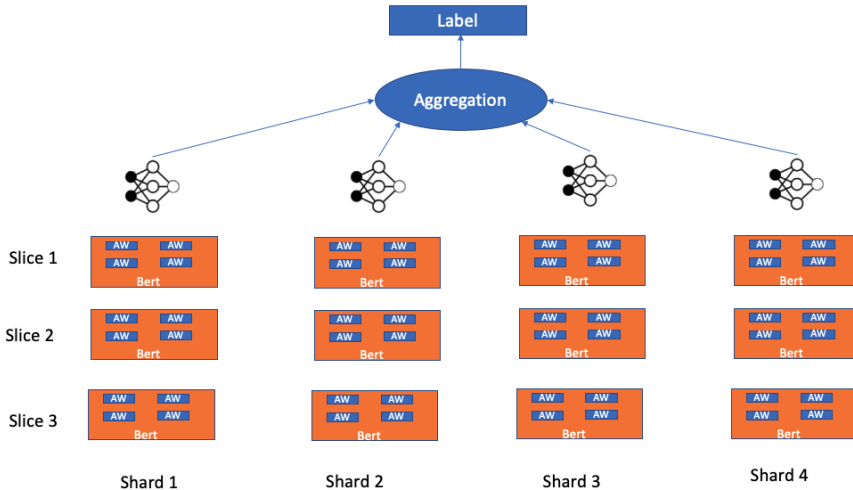
Figure 1: Architecture of SISA-A which uses S slices and R shards. One model is built for each shard and the labels are aggregated using a majority voting strategy.

Dataset Name	BERT-A		SISA-A		Accuracy Reduction (%)	Re-training time gain (%)
	Accuracy	Re-training time (s)	Accuracy	Re-training time (s)		
SST	0.93	1580	0.92	147	1	9748
QQP	0.82	1750	0.80	145	2	1106
MNLI	0.75	1868	0.70	145	6	1188

Table 1: We see that our SISA-A (SISA with Adapters) approach is only $1 - 6\%$ worse when compared to the deletion baseline model (BERT-A BERT model trained using Adapters) but has a 100x improvement in training time over the deletion baseline. This performance is measured upon receiving 16 unlearning requests

occupy on disk, we pick the first $60,000$ samples across each datasets and use 20% of this as our test split. Having the same number of training and evaluation samples helps us to compare training times across different types of NLP tasks. For each of the datasets, we use accuracy to measure model performance. The tasks are explained below:

SST-2: This is a movie review corpus from Socher et al. (2013) that consists of a sentence form the review along with the sentiment associated with that sentence.

QQP: Quora Question Pair corpus (Wang et al., 2018) consists of two questions and a label indicating if the two questions are duplicates of each other.

MNLI: The MNLI (Williams et al., 2017) corpus is built of top of SNLI. This consists of a premise, hypothesis and a label indicating if the premise and hypothesis are in entailment, contradiction or they are neutral.

4 Models

The simplest way to make a model forget the datapoint it has seen during training is to remove the datapoint from the training set and re-train the model.

However, this is computationally very expensive for large models like BERT (Devlin et al., 2018b) and we need an efficient way to re-training models if we want to forget a datapoint. Bourtoule et al. (2019) presented Sharded, Isolated, Sliced and Aggregated (SISA) training approach in order to "un-learn" a datapoint. While they measured the performance of this approach on computer vision datasets, no work has looked at its performance on NLP datasets. We explain the algorithm and our approach to make it parameter efficient for NLP specific models.

4.1 SISA

Figure 1 shows the working of the SISA algorithm. The entire training dataset is split into S shards. Each shard is made up of R slices. There is one model that is trained for every shard. In order to begin training, we can pick any ML model and then use gradient descent to train the model. While training, the model goes through the data, slice by slice saving a checkpoint after training for each slice. Finally once the model finishes training on the final slice, the model is saved and mapped to the shard. This process is continued for all the shards. During inference, each model belonging
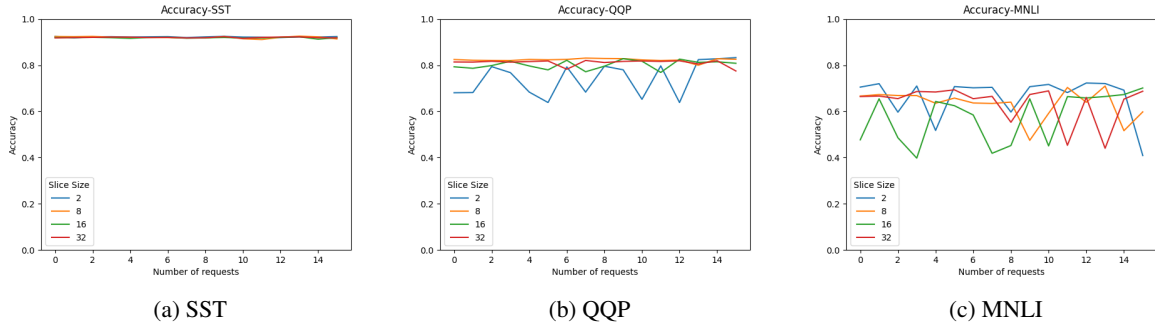
| (a) SST | (b) QQP | (c) MNLI |

Figure 2: Performance of SISA-A on the 3 datasets. Number of slices is shown in the legend. We see SISA-A has same or slightly lower performance than the baseline BERT model.
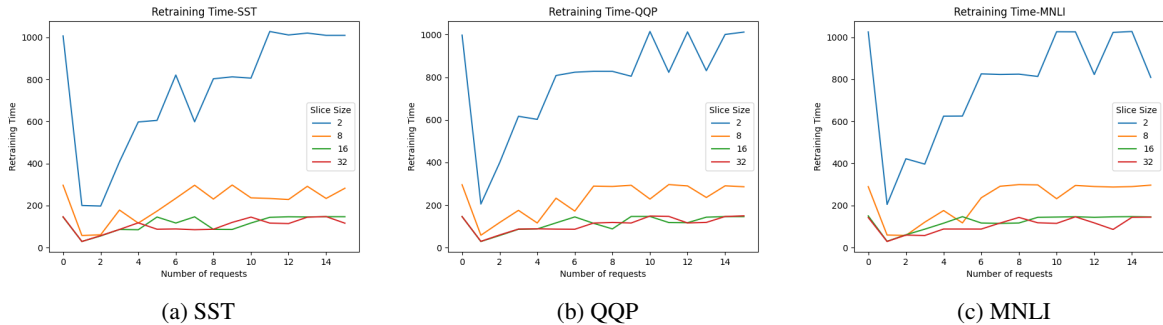

| (a) SST | (b) QQP | (c) MNLI |

Figure 3: Re-training time of SISA-A on the 3 datasets. Number of slices is shown in the legend. We see that the re-training time remains fairly constant if the slice size is greater than 8 for a number of un-learning requests.

to a shard predicts the label and the labels are aggregated similar to model ensembling. When an un-learning request comes in, we pick the shard where this data point is present, then go to the slice that contains the un-learning request. We delete the datapoint from this slice, take the checkpoint that was trained up until that checkpoint and continue training the model. This guarantees that the model forgets the un-learning request. During inference we pass the sample through each of the S models, obtain S labels and aggregate using a majority voting strategy.

4.2 SISA modified with fully connected Layer (SISA-FC)

While the SISA framework is flexible to be used with any model, in the NLP domain it is not practical to store a checkpoint after each slice. Moreover the pre-training/fine-tuning time of the full model will become a major shortcoming and we need a way to reduce both the training time and the memory footprint of these large models. A simple way to alleviate this short-coming is to use a base model and pre-train it on a generic corpus of text and then add fully connected layers on top of it. Only the pa-

rameters from the linear layers are fine tuned in the optimization process. This will reduce the overall training time as the backpropagation of gradients only happens in the final layers and also we will only need to store the weights of these additional parameters.

4.3 SISA modified with Adapters (SISA-A)

While adding linear layers to SISA might solves both the training time and the memory footprint issue, the raw performance of the model will take a hit when compared to fine tuning the entire model as is done in a regular NLP setting (Devlin et al., 2018b). One approach to keep the benefits of the linear layer while also optimizing for performance is to use Adapters (Houlsby et al., 2019) in the Encoder blocks of the transformer. While this increases the memory footprint of the model, it only accounts for about $1 - 5\%$ of the model parameters. Thereby providing us with $95 - 99\%$ memory benefits.

4.4 Baselines:

We compare the performance of SISA-FC and SISA-A with respect to popular NLP settings. We
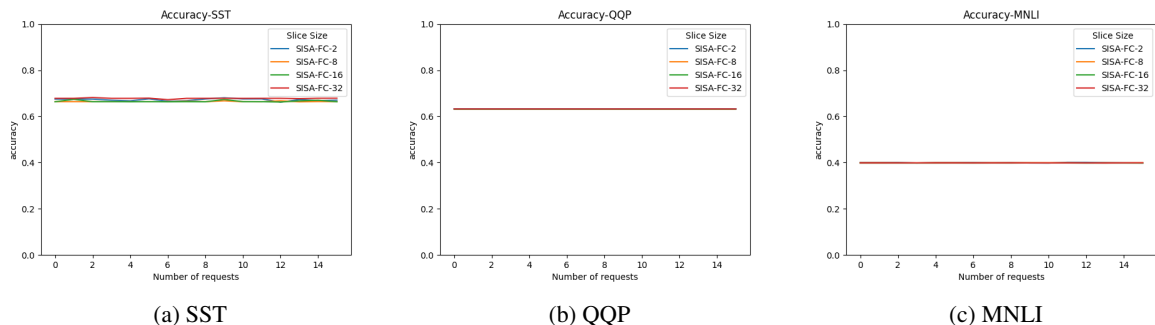
| (a) SST | (b) QQP | (c) MNLI |

Figure 4: Performance of SISA-FC on the 3 datasets. Number of slices is shown in the legend. We see the performance of SISA-FC is lower than that of SISA-A.
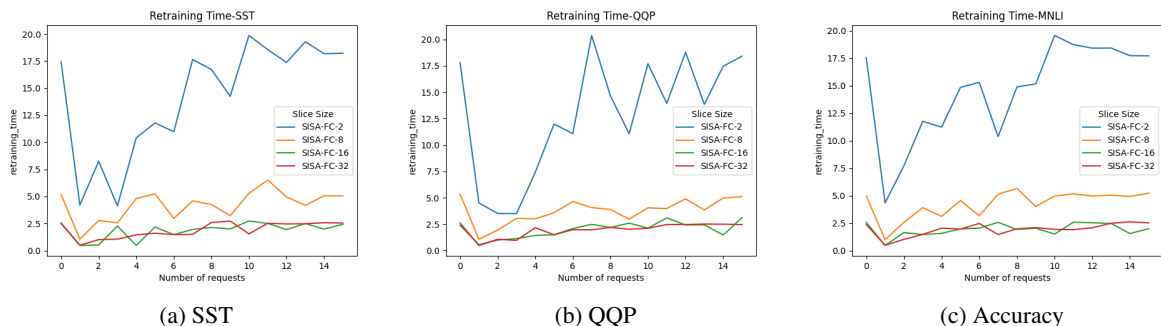

| (a) SST | (b) QQP | (c) Accuracy |

Figure 5: Re-training time of SISA-FC on the 3 datasets. Number of slices is shown in the legend. We see that the re-training time remains fairly constant if the slice size is greater than 8 for a number of un-learning requests.

fine-tune Bert on the same samples and show that there is minimal impact on performance. We also show the majority classifier accuracy to show the improvement of SISA-FC and SISA-A against this baseline.

4.5 Implementation details:

We use the Bert-base model from Huggingface[2] to train on this task. We used a batch of 16, max length of 256, Adam (Kingma and Ba, 2014) optimizer learning rate of $5e - 3$ and train the model for 10 epochs when we are training the SISA-FC model. We also use the same hyper-parameters for SISA-A approach but add adapters using Adapter-hub[3] and only update those parameters. We experiment with different slice sizes but keep the number of shards fixed at 5. We ran each experiment 4 times and averaged the results. All the experiments were run on 4 V-100 GPUs with 16GB memory each for a total of 10 days.

---

5 Experiments and Results

For all the experiments in this paper we use a shard size of 5. We however experiment with different slices sizes (these are shown in the legends of the graphs)

5.1 Evaluation Metrics:

In order for an organization to allow customers to opt-out we need mechanisms that will enable re-training of models with low re-training time and memory foot-print while keeping the model performance as close to the original model as possible. Hence we look at the following three metrics :

Accuracy: Percentage of samples in the test set that the model predicts accurately.

Re-training time: The amount of time taken to delete the un-learning requests from the dataset and then re-train the model.

Memory: The amount of memory the final model takes up on disk.

We analyze the model across all the three metrics for upto 16 uniformly randomly sampled un-learning requests unless stated otherwise. An un-learning request is defined as a request to the delete a datapoint from the training set. This is equivalent

(a) Pareto Distribution

(b) Slices affected with Pareto in orange

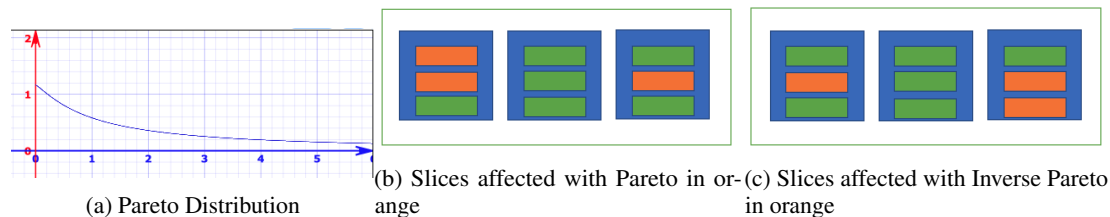(c) Slices affected with Inverse Pareto in orange

Figure 6: Figure 6a shows the Pareto distribution and the subsequent figures shows how the slices get affected with the Pareto and the inverse Pareto distributions.

to an user opting out of data collection practices.

5.2 Analysis of SISA-A

Figure 2 shows the performance of SISA-A for different slice sizes. We see that our approach is able to achieve the same level of performance as the BERT-base model even with the new setup we have with Adapters and SISA 1. While it doesn't drop in accuracy by more than $1 - 6\%$, we see that the model has a lower training time of 100x. We also see that the memory occupied by the model is much lower than using the original SISA approach which would store the weights of the full model. We also see from Figure 2 that the accuracy of our approach for slice size greater than 2 is almost the same across different slices and remains same for different number of un-learning requests as well. However with larger number of slices we see that the model has a much lower re-training time when compared to a slice size of two from Figure 3. This observation is consistent across all the 3 datasets. It is also to be noted that the re-training time does not increase linearly with more number of un-learning requests. The re-training time does increase with a positive delta. We show in Section 5.5 on how we can make this re-training time flat across multiple un-learning requests. We also see that the model re-training time is constant across all the 3 datasets that were chosen.

5.3 Analysis of SISA-FC

Figure 5 shows the performance of SISA-FC for different slice sizes. We observe that the model performance for different un-learning requests is lower than that of SISA-A by $20 - 30\%$ on all the tasks. This is due to the fact that there are far lesser number of weights that were used to fine-tune the model. However in low memory settings this approach would work better than the SISA-A approach as it occupies lesser memory. We also observe that the re-training time to just train the final layer is much lower than that of SISA-A. But

other observations made with respect to SISA-A still holds with SISA-FC.

5.4 Comparison Between SISA-FC and SISA-A

From Figures 5, we see that the SISA-A approach significantly outperforms the SISA-FC approach in terms of accuracy. Across different slice sizes we see that we can get an absolute gain of 20-30% in accuracy depending on the task. While the Houlsby et al. (2019) does note this performance difference, the performance difference is only 1%. When we apply adapters to SISA we see that the performance gain is much higher.

5.5 Profiling requests based on probability of occurrence

In the above two experiments, we assumed that the un-learning requests are uniformly randomly chosen from all of the training points. However, in a practical setting this might not be the case. Organizations can group customers into high risk and low risk customers based on their probability of opting out. In order to simulate such a scenario, we use the Pareto distribution. Pareto distribution is defined using the equation below:

$$p(x) = a.m^a/x^{a+1} \qquad (1)$$

When we plot the Pareto distribution we get the curve shown in Figure 6a. 80% of the mass is in the head of the distribution and 20% in the tail. When we sample requests based on this distribution we get data-points belonging to the shards shown in Figure 6b. For the purpose of this experiment we set m to 1 and a to 1.16. However if we take the mirror image of the Pareto distribution and sample from that distribution then the shards that will be affected is shown in Figure 6c. We experiment with both these types of sampling to see its affect on model performance and re-training time.
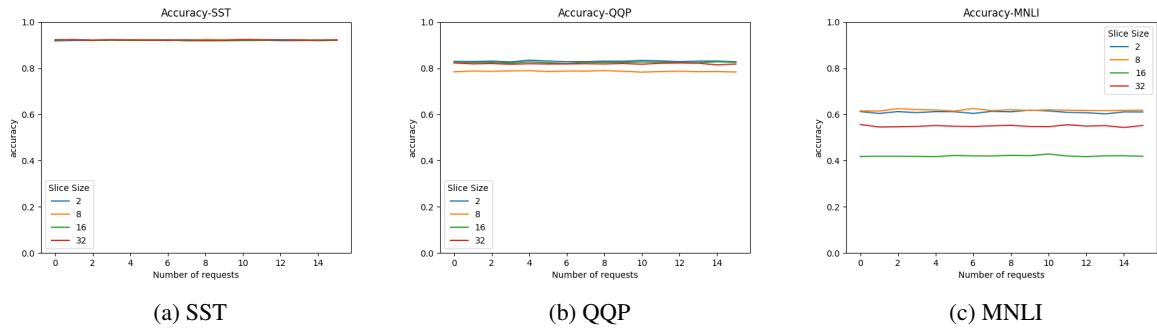
273

Figure 7: Accuracy of SISA-A on Inverse Pareto distribution. Number of slices is shown in the legend.
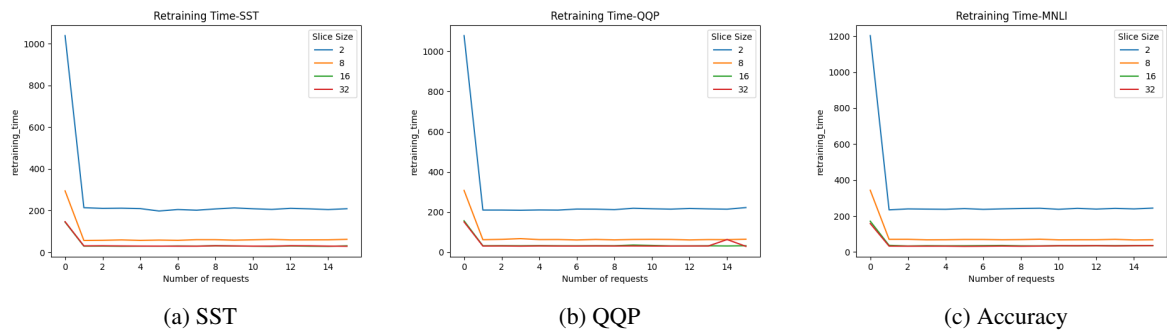

Figure 8: Re-training time of SISA-A on Inverse Pareto distribution. Number of slices is shown in the legend.
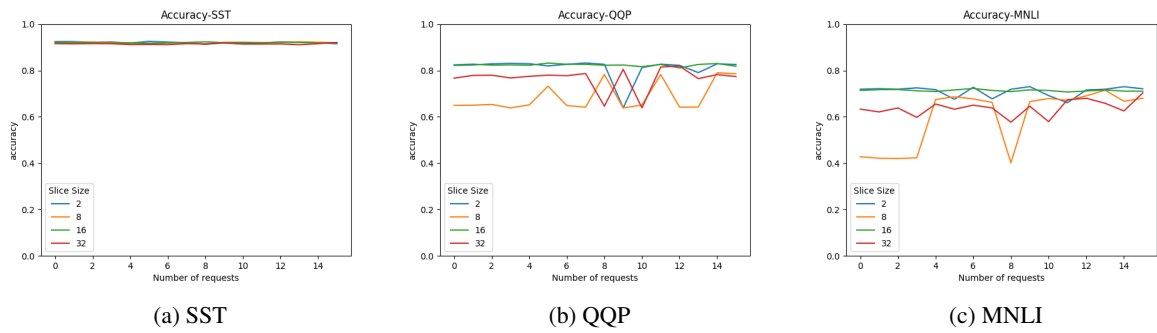

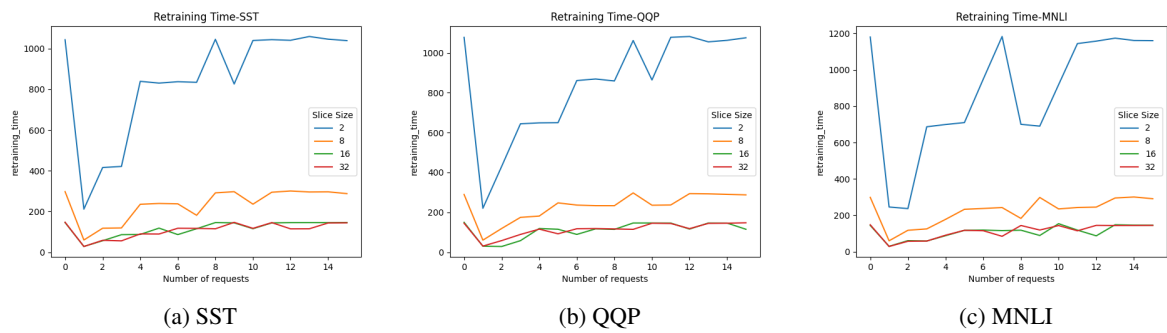Figure 9: Accuracy of SISA-A on Pareto distribution. Number of slices is shown in the legend.

Figure 10: Re-training time of SISA-A on Pareto distribution. Number of slices is shown in the legend.
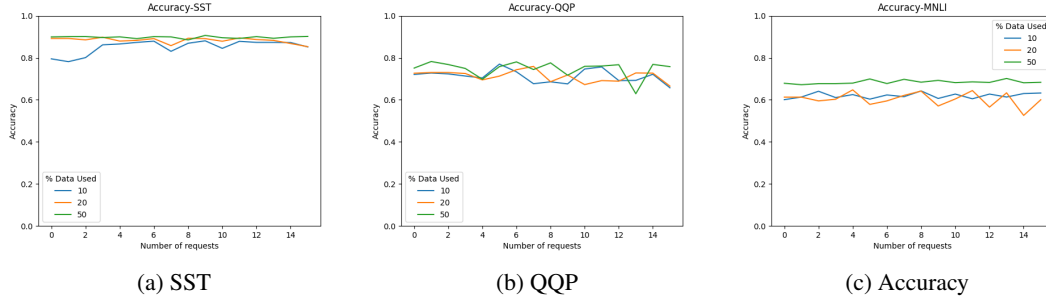
(a) SST   (b) QQP   (c) Accuracy

Figure 11: Accuracy of SISA-A on partial datasets. Percentage data used is shown in the legend

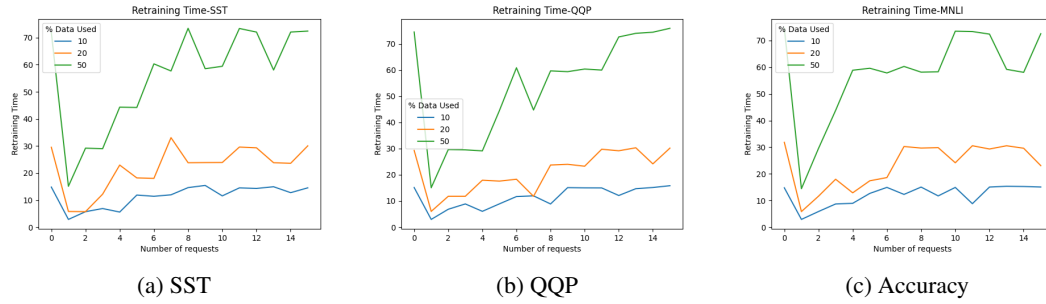
(a) SST   (b) QQP   (c) Accuracy

Figure 12: Re-training time of SISA-A on partial datasets. Percentage data used is shown in the legend

From Figures 8 and 10, we see that the training time of the inverse Pareto distribution is lesser than that of both the uniform distribution and Pareto distribution by 350% and 360% respectively. We also see that the re-training time of the inverse Pareto distribution is flat across multiple un-learning requests as they only affect the last few slices in the shard, thereby lowering training time. This opens up to possibilities of smarter customer segmentation while training. While we show one simple way to segment customers to gain on re-training time, we leave the possibilities of exploring other such strategies for future work.

5.6 Performance on a portion of the data

In the next experiment, we look at the performance of SISA-A on portions of the training set to test the few shot generalizability of this approach. For the purpose of this experiment, we use 5 shards, 16 slices and uniform un-learning requests. Results are shown in Figures 11 and 12. We see that on all the different datasets, the performance in the few-shot setting does not match the performance of the model in the full-data setting. This is one of the major drawbacks of the SISA algorithm in low resource NLP settings. Since the model is forced to look at a portion of the data several times before it is check-pointed, the model tends to overfit on

the slice it is training on. We hypothesize this to be the reason for lower model performance.

We also see that the BERT-base model performs much better than the SISA-A approach in the few-shot setting. While we show the short-coming of this approach in the few-shot setting we leave the further exploration to alleviate this issue for future work.
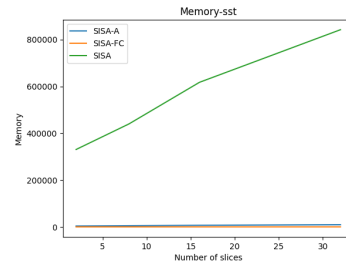

Figure 13: Memory Occupied by the three approaches. We see that the SISA-A and SISA-FC approaches provides a significant memory savings when compared to the base SISA approach

5.7 Memory profiling

Figure 13 shows the memory occupied by the three different approaches compared in this work. We see that if we use the BERT model along with the vanilla SISA algorithm, the memory increases linearly as the number of slices increases. But

with our approach, the memory increase is far less drastic (SISA-A and SISA-FC).

6 Conclusion

The central tenet of privacy regulations in the U.S and EU revolve around the concept of "notice and choice". This allows users to opt-out of data collection if they deem something violates their privacy needs. However, owing to the gap between technology experts and privacy regulators there has not been enough work around opting out of data that has already been used to train ML models. Through the SISA-A and SISA-FC approaches we show the working of the SISA algorithm in the NLP domain on Encoder based models. We show the benefits of our approach in terms of re-training time and disk space usage.

Limitations

While the experimental results show that it is possible to use SISA in LLMs, it is only limited to Encoder based models because of the requirement of an aggregation layer in SISA. We cannot extend this approach to the decoder based models as is and that is a limitation of our work. Also,as noted in Section 5.6, our results show that SISA-A approches do not work well in the few shot setting because of the tendency of the model to overfit on each slice. In this paper we do point out the lack of generalizability of this approach in the few shot setting. However, we do not discuss steps to mitigate these issues using our technique. Finally, we show how organizations can benefit from segmenting users into different groups based on the probability of opting out. While we show this by using/simulating different distributions we do not perform profiling on actual user data.

Ethics Statement

In this work we show how we can perform efficient NLP while also adhering to the people's privacy requirements. While we show how organizations can efficiently perform the task of un-learning we cannot prove if the organization indeed performed unlearning on the samples. Also all the associated risks of using pre-trained language models like BERT will remain with this approach as they form the backbone of our framework.

References

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

Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. 2020. Machine unlearning: Linear filtration for logit-based classifiers.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2019. Machine unlearning. *CoRR*, abs/1912.03817.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. 2019. Lifelong anomaly detection through unlearning. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1283–1297, New York, NY, USA. Association for Computing Machinery.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. 2019. Eternal sunshine of the spotless net: Selective forgetting in deep networks. *CoRR*, abs/1911.04933.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2022. Knowledge unlearning for mitigating privacy risks in language models.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *CoRR*, abs/2110.05679.

Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N. Ravi. 2022. Deep unlearning via randomized conditionally independent hessians.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Variational bayesian unlearning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16025–16036. Curran Associates, Inc.

Quoc Phong Nguyen, Ryutaro Oikawa, Dinil Mon Divakaran, Mun Choon Chan, and Bryan Kian Hsiang Low. 2022a. Markov chain monte carlo-based machine unlearning. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. ACM.

Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022b. A survey of machine unlearning.

Anand D. Sarwate, Kamalika Chaudhuri, and Claire Monteleoni. 2009. Differentially private support vector machines. *CoRR*, abs/0912.0071.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Aman Tahiliani, Vikas Hassija, Vinay Chamola, and Mohsen Guizani. 2021. Machine unlearning: Its need and implementation strategies. In *2021 Thirteenth International Conference on Contemporary Computing (IC3-2021)*, IC3 '21, page 241–246, New York, NY, USA. Association for Computing Machinery.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2021. Differentially private fine-tuning of language models. *CoRR*, abs/2110.06500.