Understanding Unintended Memorization in Language Models Under Federated Learning

Om Thakkar and **Swaroop Ramaswamy** and **Rajiv Mathews** and **Françoise Beaufays**
Google LLC,
Mountain View, CA, U.S.A.
{omthkkr, swaroopram, mathews, fsb} @google.com

Abstract

Recent works have shown that language models (LMs), e.g., for next word prediction (NWP), have a tendency to memorize rare or unique sequences in the training data. Since useful LMs are often trained on sensitive data, it is critical to identify and mitigate such *unintended* memorization. Federated Learning (FL) has emerged as a novel framework for large-scale distributed learning tasks. It differs in many aspects from the well-studied *central learning* setting where all the data is stored at the central server, and minibatch stochastic gradient descent is used to conduct training. This work is motivated by our observation that NWP models trained under FL exhibited remarkably less propensity to such memorization compared to the central learning setting. Thus, we initiate a formal study to understand the effect of different components of FL on unintended memorization in trained NWP models. Our results show that several differing components of FL play an important role in reducing unintended memorization. First, we discover that the clustering of data according to users—which happens by design in FL—has the most significant effect in reducing such memorization. Using the Federated Averaging optimizer with larger effective minibatch sizes for training causes a further reduction. We also demonstrate that training in FL with a user-level differential privacy guarantee results in models that can provide high utility while being resilient to memorizing *out-of-distribution* phrases with thousands of insertions across over a hundred users in the training set.

1 Introduction

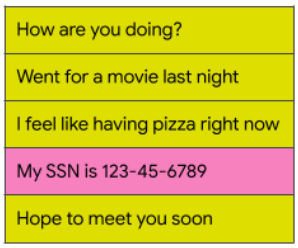
There is a growing line of work (Fredrikson et al., 2015; Wu et al., 2016; Shokri et al., 2017; Carlini et al., 2018; Song and Shmatikov, 2019) demonstrating that neural networks can leak information about the underlying training data in unexpected ways. Many of these works show that language models (LMs), which include commonly-used next word prediction (NWP) models, are prone to *unintentionally memorize* rarely-occurring phrases in the data. Large-scale LM training often involves training over sensitive data, and such memorization can result in blatant leaks of privacy (e.g., (Munroe, 2019)). Thus, it is crucial to measure such memorization in trained LMs, and identify mitigation techniques to ensure privacy of the training data.

The framework of Federated Learning (FL) (McMahan et al., 2017a; McMahan and Ramage, 2017) has emerged as a popular approach for training neural networks on a large corpus of decentralized on-device data (e.g., (Konečný et al., 2016; Konecný et al., 2016; Bonawitz et al., 2017; Hard et al., 2018; Bonawitz et al., 2019)). FL operates in an iterative fashion: in each round, sampled client devices receive the current global model from a central server to compute an update on their locally-stored data, and the server aggregates these updates using, for e.g., the Federated Averaging (FedAvg) algorithm (McMahan et al., 2017a), to build a new global model. A hallmark of FL is that each participating device only sends model weights to the central server; raw data never leaves the device, remaining locally-cached. This, by itself, is not sufficient to provide formal privacy guarantees for the training data. However, this work is motivated by the observation (described in detail in Section 3) that NWP models trained under the canonical setting of FL exhibited resilience to memorize rare phrases in spite of hundreds of occurrences in the training data. Note that FL does differ in many aspects from the well-studied (Shokri et al., 2017; Carlini et al., 2018; Song and Shmatikov, 2019) *central learning* setting where all the data is stored at a central server, and minibatch stochastic gradient descent (SGD) is used to conduct training. While training NWP models via central learning, we observed that phrases with even tens of occurrences were easily

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1–10

https://doi.org/10.26615/978-954-452-056-4_001

(a) A user selected as a secret sharer for a canary.



(b) An example in a secret sharer's local dataset being replaced by the canary.

Figure 1: An illustration of our federated secret-sharer framework, using the canary "My SSN is 123-45-6789".

memorized, in line with prior work (Carlini et al., 2018) that showed the propensity of such models to memorize phrases with even one occurrence in the training set. Thus, we initiate a formal study to understand the effect of the different components of FL, compared to the central learning setting, on unintended memorization in trained NWP models.

We also study the extent to which a guarantee of Differential Privacy (DP) (Dwork et al., 2006c,a) reduces such memorization. DP has become the standard for performing learning tasks over sensitive data, and has been adopted by companies like Google (Erlingsson et al., 2014; Bittau et al., 2017; Erlingsson et al., 2020), Apple (Apple, 2017), Microsoft (Ding et al., 2017), and LinkedIn (Rogers et al., 2020), as well as the US Census Bureau (Kuo et al., 2018). Intuitively, DP prevents an adversary from confidently making conclusions about whether any particular user's data was used to train a model, even while having access to the model and arbitrary external side information.

The Federated Secret Sharer: We build on the "secret sharer" framework (Carlini et al., 2018) that was designed to measure the unintended memorization in generative models. At a high-level, *out-of-distribution* examples (called canaries) are inserted into a training corpus, and a model trained on this corpus is then evaluated using various techniques to measure the extent to which the model has *memorized* the canaries. Since datasets in FL are inherently partitioned according to users, we adapt the secret sharer framework to the FL regime by introducing two parameters to control the presence of a canary in such settings. An illustration of our federated secret sharer framework is shown in Figure 1. Given a canary with parameters p_u and p_e, we let p_u be the probability with which each user in a dataset is selected to be a "secret sharer" of the canary (Figure 1a), whereas p_e denotes the probability with which each example in such a secret sharer's data is replaced by the canary (Figure 1b). We use Poisson sampling for both user-selection and example-replacement. The secret sharer selection phase precedes canary insertion to model real-world settings where occurrences of user-specific unique or rare out-of-distribution canaries are typically limited to a small group of users, but such users can exhibit high usage for those canaries.

Contributions: Our empirical evaluations demonstrate the following key contributions. First, we observe that clustering training data according to users, which happens by design in distributed learning settings like FL, has a significant effect in reducing unintended memorization for NWP models. Next, given a dataset partitioned by users, we show that replacing the learning optimizer from SGD to Federated Averaging and increasing the effective minibatch size provides a further reduction in such memorization. Lastly, we demonstrate that training in FL with *user-level* differential privacy (DP) results in models that can provide comparable utility while being resilient to memorizing canaries with thousands of insertions spread across over a hundred users in the training set. Prior work (Carlini et al., 2018) has shown that models trained with *record-level* DP do not exhibit unintended memorization for a single insertion of a canary. We provide evidence of models being resilient to memorizing canaries for orders of magnitude higher insertions, at the stronger user-level privacy.

1.1 Related Work

Apart from (Carlini et al., 2018) which this work builds upon, other works (Song and Shmatikov, 2019) have also studied memorization in generative text models. The FL paradigm, which is a major

focus of this work, has been used to train multiple production scale models (Hard et al., 2018; Ramaswamy et al., 2019; Chen et al.). Kairouz et al. (2019) provides an excellent overview of the state-of-the-art in the field, along with a suite of interesting open problems. This work also studies the effectiveness of a user-level DP guarantee in reducing unintended memorization. While many works on DP focus on *record-level* DP guarantees (which usually cannot be directly extended to strong user-level DP guarantees), recent works (e.g., (McMahan et al., 2017b; Jain et al., 2018; Augenstein et al., 2020; Andrew et al., 2021)) have designed techniques tailored to user-level DP guarantees.

2 Contrasting Federated Learning with Central Learning

Now, we take a deeper look at how the well-studied central learning framework differs from the canonical setting of FL for LM training. We are interested in differences that might have an effect on unintended memorization. We identify three such components: (1) *Data Processed per Update:* Central learning typically ingests data as *records*/sentences. On the other hand, FL operates at the granularity of a *user*, with each user having their own set of sentences locally. Typically, the amount of data processed per model update in central learning is much smaller in comparison to FL. (2) *Learning Technique:* In central learning, the model is updated via SGD on a minibatch of records. In the canonical setting of FL, a model update typically corresponds to Federated Averaging over a minibatch of users: an average of the differences between the current model and the model obtained after several SGD steps on the local data of a user. (3) *Independent and Identically Distributed (IID) Data:* To reduce variance in learning, the data in central learning is shuffled before training (and/or each update involves a randomly sampled minibatch). Thus, each minibatch can be estimated to be drawn IID from the data. Datasets in FL are naturally grouped according to potentially heterogeneous users, resulting in non-IID data even though each minibatch of users may be randomly sampled.[1]

[1]We do not discuss unbalanced datasets, i.e., the fact that users can have varying amounts of local data, since Federated Averaging in FL deals with such imbalances by weighing each client update according to the size of its local data.

3 Empirical Evaluation

Experimental Setup: Our model architecture (1.3M parameters) mirrors the one used in Hard et al. (2018). We create a modified version of the Stack Overflow dataset (Overflow, 2018) hosted by TensorFlow Federated (Ingerman and Ostrowski, 2019), containing 392K users (93M records). For an IID version of this dataset, we randomly shuffle all the records, and create *synthetic* users having data assigned sequentially from the shuffled records. Since our model is a word-level language model, we follow the methodology used by Carlini et al. (2018) for their experiments with the GMail Smart Compose model (Chen et al., 2019). We insert random 5-word canaries with configurations in the cross product of $p_u \in \{1/50\text{K}, 3/50\text{K}, 1/5\text{K}\}$ and $p_e \in \{1\%, 10\%, 100\%\}$, with 10 different canaries for each (p_u, p_e) configuration, resulting in the insertion of 90 different canaries. Given a prefix of a canary, we use two methods to evaluate the unintended memorization of the suffix for a model: Random Sampling (RS), which for a 2-word prefix measures if the canary has the least log-perplexity among 2M random suffixes, and Beam Search (BS), which uses a greedy beam search to see if the canary is in its top 5 most-likely 5-word continuations from a 1-word prefix. We measure the utility of a model with accuracy and perplexity on the test partition of the unmodified Stack Overflow dataset.

Empirical Results: We present the results of our experiments on evaluating unintended memorization under different training regimes ranging from canonical FL to central learning. For all our experiments using SGD, we train models for 37.5M steps, whereas we train for 8000 rounds for the experiments using FedAvg. For the largest minibatch sizes used in both settings (256 records for SGD, and 5000 users for FedAvg), these checkpoints correspond to training for 100 epochs. Table 1 shows the number of canaries (out of 90) that show up as memorized via both the RS and BS methods. The utility of all the evaluated models is similar; accuracy varies from $23.7 - 24.6\%$, and perplexity varies from $57.3 - 64.3$ across all models.[2]

Training in FL with DP Federated Averaging (DP-FedAvg): Next, we evaluate the extent to which training using DP-FedAvg is resilient to such

[2]We defer the utility measurements to Table 3.

Optimizer	Data	Batch Size	RS	BS
FedAvg	Non-IID	500 users	21	0
		1K users	23	1
		2K users	19	1
		5K users	26	2
	IID	500 users	66	56
		1K users	69	58
		2K users	67	56
		5K users	65	58
SGD	Non-IID	32 records	37	19
		64 records	49	36
		128 records	48	34
		256 records	51	39
	IID	32 records	54	42
		64 records	54	42
		128 records	52	45
		256 records	53	43

Table 1: Results for the number of inserted canaries (out of 90) memorized via the Random Sampling (RS) and Beam Search (BS) methods for various models evaluated at 8000 rounds when sampling users (FedAvg), and 37.5M steps when sampling records (SGD).

memorization. To provide the strongest user-level DP while obtaining high utility, we conduct experiments only for our largest minibatch size of 5K users. The results are presented in Table 2.

Optimizer	RS	BS	Acc. %	Perp.
FedAvg	26	2	24.5	58.2
DP-FedAvg	12	0	23.3	68.5

Table 2: Unintended memorization and utility for a model trained with $(18.8, 10^{-7})$-DP in FL (non-IID users) with 5k users/round for 100 epochs.

3.1 Discussion

Clustering data according to users: The results from our experiments strongly indicate that clustering data according to users significantly reduces unintended memorization. This is evident by considering the measurements in Table 1 in pairs where the only differing component among them is whether the data is IID or not. The number of epochs taken over the dataset to train the models on which we measure memorization is the same for any particular minibatch size, irrespective of whether the data is IID. Thus, the number of times the inserted canaries were encountered during training is the same. However, the amount of memorization observed is always lower when the data is Non-IID. This effect is more pronounced in the settings where FedAvg is used as the training method. For instance, for a minibatch size of $b_u = 500$ users, training with FedAvg on IID data results in 66 (56) canaries showing up as memorized via the RS (BS) method. The same configuration on Non-IID data results in the RS method classifying only 21 canaries as memorized, and the BS method not being able to extract any canary even after 8000 rounds of training. In addition to the data being clustered, the inserted canaries are clustered as well, which we conjecture to be playing a crucial role in reducing such memorization.

Varying data per update: Fixing the optimizer to SGD/FedAvg and the data to be IID/non-IID, we do not see any significant effect of varying the minibatch size on such memorization.[3]

Training non-IID user data with FedAvg and larger effective minibatches: The smallest minibatch size for our FedAvg experiments is 500 users,[4] and as each user contains ≈ 250 records, the *effective* minibatch size is ≈ 125K records. In comparison, the largest minibatch size for which we are able to conduct SGD training is 256 records. Focusing on the results in Table 1 using Non-IID data, we find that using FedAvg and having larger effective minibatches per round causes a significant reduction in unintended memorization when compared to training with SGD and smaller minibatches.[5]

Training with DP-FedAvg in FL: Our aim is to test the extent to which NWP models trained with DP-FedAvg in FL are resilient to such memorization. By definition, a user-level DP guarantee is intended to be resilient to changes w.r.t. any one

[3] For the case of SGD and Non-IID data, while unintended memorization does seem to increase with the minibatch size, we do not observe the increase consistently. Additional investigation for potential causes of such a trend are beyond the scope of this work.

[4] We do not train with smaller user minibatch sizes as we want the number of training epochs for the lowest settings in FedAvg and SGD to be similar at 8000 training rounds.

[5] Looking at the same set of results for IID data, the trend seems to be moving in the direction of increasing memorization. However, since the magnitude of the effect is much smaller, we deem that further investigation is required for this case, which we leave for future work.

user's data. Some of our inserted canaries are shared by $\sim$100 users (with $\sim$24.5K occurrences in the training data). In spite of such high levels of canary insertion, and our FL models exhibiting the least amount of unintended memorization (Table 1), we see that training with DP-FedAvg results in a significantly reduced memorization. Our results are noteworthy as, in spite of our DP model exhibiting extremely low unintended memorization, it also provides comparable utility as a model trained via FedAvg, along with a user-level guarantee of $(18.8, 10^{-7})$-DP. While strengthening the privacy guarantee of DP-FedAvg by increasing the noise added to the model update in each training round can further reduce such memorization, it can also start significantly affecting model utility. Designing methods that improve the privacy-utility trade-offs is an interesting direction, which is beyond the scope of this work.

4 Conclusion

In this work, we conduct a formal study to understand the effect of the different components of Federated Learning (FL), on the unintended memorization in trained next word prediction (NWP) models, as compared to the well-studied central learning. From our results, we observe that the components of FL exhibit a synergy in reducing such memorization. To our surprise, user-based clustering of data (which occurs as a natural consequence in the FL setting) has the most significant effect in the reduction. Moreover, training using Federated Averaging and larger effective minibatches reduces such memorization further. Lastly, we observe that training in FL with a user-level differential privacy guarantee results in models that can provide comparable utility while being resilient to memorizing canaries with thousands of insertions across over a hundred users in the training set.

Recent work (Karimireddy et al., 2019) has shown that, in general, such heterogeneity in the training data can result in a slower and unstable convergence due to factors such as "client-drift". For all of the experiments with non-IID data, we observe that the utility of the trained models is comparable to those trained on IID data, and we leave further exploration into why client-drift may not play a significant role in our experiments for future work. Next, while our extensive evaluation is for a practical NWP model on a real-world benchmark dataset, the degree of unintended memorization in general can depend on the model architecture and the dataset used for training. Lastly, the secret-sharer line of methods for measuring unintended memorization operate at the granularity of a record. For future work, it will be interesting to design stronger attacks targeting data at the granularity of a user, and measure the resilience of models trained via FL, against such memorization.

Acknowledgements

The authors would like to thank Mingqing Chen, Andrew Hard, and Gautam Kamath for their helpful comments towards improving the paper.

References

Galen Andrew, Om Thakkar, H. Brendan McMahan, and Swaroop Ramaswamy. 2021. Differentially private learning with adaptive clipping.

Differential Privacy Team Apple. 2017. Learning with privacy at scale.

Sean Augenstein, H. Brendan McMahan, Daniel Ramage, Swaroop Ramaswamy, Peter Kairouz, Mingqing Chen, Rajiv Mathews, and Blaise Agüera y Arcas. 2020. Generative models for effective ML on private, decentralized datasets. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. 2017. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 441–459. ACM.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 Association for Computing Machinery (ACM) Special Interest Group on Security, Audit and Control (SIGSAC) Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA. ACM.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. 2018. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *Computing Research Repository (CoRR)*, abs/1802.08232.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295.

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. Federated learning of n-gram language models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, November 3-4, 2019*.

Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3571–3580.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006a. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503.

Cynthia Dwork, Krishnaram Kenthapadi, Frank Mcsherry, Ilya Mironov, and Moni Naor. 2006b. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503.

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

Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. 2020. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *CoRR*, abs/2001.03618.

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 Association for Computing Machinery (ACM) SIGSAC conference on computer and communications security*, pages 1054–1067. Association for Computing Machinery (ACM).

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd Association for Computing Machinery (ACM) SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA. Association for Computing Machinery (ACM).

Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604.

Alex Ingerman and Krzys Ostrowski. 2019. Introducing tensorflow federated.

Prateek Jain, Om Thakkar, and Abhradeep Thakurta. 2018. Differentially private matrix completion revisited. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 2220–2229.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and open problems in federated learning. *CoRR*, abs/1912.04977.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2019. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. *Computing Research Repository (CoRR)*, abs/1610.05492.

Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *ArXiv*, abs/1610.02527.

Yu-Hsuan Kuo, Cho-Chun Chiu, Daniel Kifer, Michael Hay, and Ashwin Machanavajjhala. 2018. Differentially private hierarchical count-of-counts histograms. *PVLDB*, 11(11):1509–1521.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017a. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282.

Brendan McMahan and Daniel Ramage. 2017. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017b. Learning differentially private language models without losing accuracy. *CoRR*, abs/1710.06963.

I. Mironov. 2017. Rényi differential privacy. In *2017 Institute of Electrical and Electronics Engineers (IEEE) 30th Computer Security Foundations Symposium (CSF)*, pages 263–275.

Randall Munroe. 2019. xkcd: Predictive models. https://xkcd.com/2169/.

Stack Overflow. 2018. The Stack Overflow Data. https://www.kaggle.com/stackoverflow/stackoverflow.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard.

Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. 2020. Linkedin's audience engagements api: A privacy preserving data analytics system at scale.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 Institute of Electrical and Electronics Engineers (IEEE) Symposium on Security and Privacy (SP)*, pages 3–18.

Congzheng Song and Vitaly Shmatikov. 2019. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 196–206. ACM.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. 2019. Subsampled renyi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 1226–1235.

X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. 2016. A methodology for formalizing model-inversion attacks. In *2016 Institute of Electrical and Electronics Engineers (IEEE) 29th Computer Security Foundations Symposium (CSF)*, pages 355–370.

A Preliminaries

A.1 Measuring Unintended Memorization

Following (Carlini et al., 2018), this work assumes a threat model of curious or malevolent users having a black-box query access to models, in that they see only the models' output probabilities (or logits). We also assume that users can adaptively query models multiple times, thus posing a threat of extracting uncommon word combinations.

Now, we describe the Secret Sharer framework of (Carlini et al., 2018). First, random sequences called canaries are inserted into the training data. The canaries are constructed based on a prefixed format sequence. For instance, to design the framework for a character-level model, the format could be "My SSN is xxx-xx-$xxxx$", where each x can take a random value from digits 0 to 9. Next, the target model is trained on the modified dataset containing the canaries. Lastly, methods like Random Sampling and Beam Search (both formally defined in Section 3) are used to efficiently measure the extent to which the model has "memorized" the inserted random canaries, and whether it is possible for an adversary with partial knowledge to extract the canary. For instance, if a canary is classified as memorized via our Beam Search method, then given black-box access to the trained model, an adversary with the knowledge of only the first word of the inserted canary can extract it completely with a simple beam search.

A.2 Differential Privacy

To establish the notion of differential privacy (Dwork et al., 2006c,b), we first define neighboring datasets. We will refer to a pair of datasets D, D' as neighbors if D' can be obtained by the addition or removal of all the examples associated with one user from D, to be able to provide a user-level DP guarantee.[6]

Definition A.1 (Differential privacy (Dwork et al., 2006c,b))**.** *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if, for any pair of neighboring*

[6]This is in contrast to a record-level DP guarantee, where neighboring datasets differ in the addition/removal of exactly one example.

datasets D and D', and for all events $\mathcal{S}$ in the output range of $\mathcal{A}$, we have

$$\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(D') \in \mathcal{S}] + \delta$$

where the probability is taken over the random coins of $\mathcal{A}$.

For meaningful privacy guarantees, ϵ is assumed to be a small constant, and $\delta \ll 1/|D|$.

To train models with DP guarantees, we follow the variant of DP Federated Averaging (DP-FedAvg) (McMahan et al., 2017b) used in (Augenstein et al., 2020), where the only change is sampling fixed-sized minibatches in each training round.[7]

A.3 Differentially Private Federated Averaging

We now present the technique used to train our DP model in FL. It closely follows the DP-FedAvg technique in (McMahan et al., 2017b), in that per-user updates are clipped to have a bounded L_2 norm, and calibrated Gaussian noise is added to the weighted average update to be used for computing the model to be sent in the next round. A slight difference between the DP-FedAvg algorithm in (McMahan et al., 2017b) and our approach is the way in which client devices are sampled to participate in a given federated round of computation. DP-FedAvg uses Poisson sampling, where for each round, each user is selected independently with a fixed probability. In this work (also, following (Augenstein et al., 2020)), we instead use fixed-size federated rounds, where a fixed number of users is randomly sampled to participate in each round. For reference, we provide a pseudo-code for the technique in Algorithm 1.

Privacy analysis: Following the analysis of this technique in (Augenstein et al., 2020), we obtain our DP guarantees by using the following:

1. the analytical moments accountant (Wang et al., 2019) to obtain the Rényi differential privacy (RDP) guarantee for a federated round of computation that is based on the subsampled Gaussian mechanism,
2. Proposition 1 (Mironov, 2017) for computing the RDP guarantee of the composition involving all the rounds, and
3. Proposition 3 (Mironov, 2017) to obtain a DP guarantee from the composed RDP guarantee.

[7] Due to a technical limitation of the simulation framework, our experiments use sampling with replacement instead of without replacement; this should have negligible impact on the metrics of the trained models.

Main training loop:
parameters: round participation fraction $q \in (0, 1]$, total user population $N \in \mathbb{N}$, noise scale $z \in \mathcal{R}^+$, clip parameter $S \in \mathcal{R}^+$

Initialize model θ^0, moments accountant $\mathcal{M}$
Set $\sigma = \frac{zS}{qW}$
for each round $t = 0, 1, 2, \ldots$ **do**
  $\mathcal{C}^t \leftarrow$ (sample without replacement qN users from population)
  for each user $k \in \mathcal{C}^t$ **in parallel do**
    $\Delta_k^{t+1} \leftarrow \text{UserUpdate}(k, \theta^t)$
  $\Delta^{t+1} = \frac{1}{qN} \sum_{k \in \mathcal{C}^t} \Delta_k^{t+1}$
  $\theta^{t+1} \leftarrow \theta^t + \Delta^{t+1} + \mathcal{N}(0, I\sigma^2)$
  $\mathcal{M}$.accum_priv_spending(z)
print $\mathcal{M}$.get_privacy_spent()

UserUpdate(k, θ^0)**:**
parameters: number of local epochs $E \in \mathbb{N}$, batch size $B \in \mathbb{N}$, learning rate $\eta \in \mathcal{R}^+$, clip parameter $S \in \mathcal{R}^+$, loss function $\ell(\theta; b)$

$\theta \leftarrow \theta^0$
for each local epoch i from 1 to E **do**
  $\mathcal{B} \leftarrow$ (k's data split into size B batches)
  for each batch $b \in \mathcal{B}$ **do**
    $\theta \leftarrow \theta - \eta \bigtriangledown \ell(\theta; b)$
$\Delta = \theta - \theta^0$
return update $\Delta_k = \Delta \cdot \min\left(1, \frac{S}{\|\Delta\|}\right)$ *// Clip*

Algorithm 1: Differentially Private Federated Averaging (DP-FedAvg) with fixed-size federated rounds, used to train our DP NWP model.

B Additional Empirical Evaluation

In this section, we present the results of our additional empirical evaluation that was omitted from the main body.

Utility Metrics: Here, we present the utility metrics for all the models for which the unintended memorization results were presented in Table 1. It is easy to see that the utility of all the evaluated models is similar; the accuracy varies from $23.7 - 24.6\%$, and the perplexity varies from $57.3 - 64.3$ across all the models.

Using Different Optimizers: In Table 4, we provide the results for using different optimizers like Momentum (Qian, 1999) and Adam (Kingma and Ba, 2015) for training. We conduct experiments using only the smallest batch size in both the granularities (32 records, or 500 users). For Momentum, we set the momentum parameter to 0.9, and for

Optimizer	Data	Batch size	Acc.	Perp.
FedAvg	Non-IID	500 users	24.4	58.8
		1K users	24.3	59.5
		2K users	24.5	58.3
		5K users	24.5	58.2
	IID	500 users	24.6	57.5
		1K users	24.6	57.3
		2K users	24.6	57.4
		5K users	24.6	57.3
SGD	Non-IID	32 records	23.7	64.3
		64 records	24.1	61.8
		128 records	24.1	61.5
		256 records	24.1	61.3
	IID	32 records	24	62.2
		64 records	24.1	61.5
		128 records	24	62
		256 records	24.1	61.1

Table 3: Results for the utility metrics for various models evaluated at 8000 rounds when sampling users (FedAvg), and 37.5M steps when sampling records (SGD). Acc. denotes test accuracy (in %), and Perp. denotes test perplexity.

Data	Batch Size	Opt.	RS, BS	Acc. (%), Perp.
IID	32 rec.	SGD	54, 42	24, 62.2
		Mom.	64, 50	24.2, 60.6
		Adam	56, 42	22.7, 70.8
	500 users	FedAvg	66, 56	24.6, 57.5
		FedAvg + Adam	60, 46	23.7, 63.7
Non-IID	32 rec.	SGD	37, 19	23.7, 64.3
		Mom.	48, 36	24.3, 59.9
		Adam	21, 13	21.5, 84.1
	500 users	FedAvg	21, 0	24.4, 58.8
		FedAvg + Adam	8, 0	23.3, 66.5

Table 4: Unintended memorization, and utility metrics for models using different optimizers evaluated at 37.5M steps when sampling records (e.g., SGD), and 8000 rounds when sampling users (e.g., FedAvg). Mom. denotes the Momentum optimizer.

Optimizer	RS	BS	Acc. %	Perp.
FedAvg	65	58	24.6	57.3
DP-FedAvg	48	42	23.9	63

Table 5: Unintended memorization and utility for models trained with $(18.8, 10^{-7})$-DP using DP-FedAvg on IID data and 5K users/round for 100 epochs.

Adam, we set the learning rate to 10^{-4}. First, we observe that using Momentum increases the observed unintended memorization but has a similar utility as SGD. On the other hand, we see that using Adam decreases such memorization, but the utility of the models is also noticeably reduced as compared to SGD. We observe a similar trend when Adam is combined with FedAvg.

Training with DP-FedAvg on IID data: Now, we present in Table 5 the results of using DP-FedAvg as the optimizer on IID data consisting of synthetic users. We observe that using DP-FedAvg provides a significant reduction in unintended memorization compared to using FedAvg.

Only Clipping: To bound the contribution by each participating user, DP-FedAvg clips each user update before aggregating them from a minibatch of users and adding calibrated noise to guarantee DP. Following (Carlini et al., 2018), we present results (rows containing "FedAvg+Clip" in Table 6) for the case when user updates are clipped to a value of 0.2, but no noise is added. This results in an (∞, δ)-DP guarantee for any $\delta \in (0, 1)$, which is vacuous as a privacy guarantee. However, this experiment helps us observe the extent to which only clipping reduces the propensity of unintended memorization exhibited by trained models. With IID data, for the setting evaluated in Section 3 (8000 rounds, i.e., 100 epochs for minibatch size of 5000 users), we observe that the RS method extracts 58 canaries with clipping, which is 7 fewer canaries when compared to without clipping. The BS method extracts 49, which is 9 fewer than without clipping. For Non-IID data, we observe a similar trend but it is more pronounced: the RS method extracts 11 canaries with clipping, which is 15 fewer canaries when compared to without clipping, whereas the BS method is not able to extract any of the inserted canaries.

Evaluating for Same Training Epochs: In Table 7, we provide the results for evaluating models trained for the same number of epochs over the training data. For the runs using SGD, we start

Data	Optimizer	RS, BS	Acc. (%), Perp.
IID	FedAvg	65, 58	24.6, 57.3
	FedAvg+Clip	58, 49	24.2, 60
Non-IID	FedAvg	26, 2	24.5, 58.2
	FedAvg+Clip	11, 0	24, 61.5

Table 6: Unintended memorization, lowest (by insertion frequency) canary configuration memorized, and utility for models trained with Clipping/DP and 5000 users/round for 100 epochs. The models trained with DP-FedAvg satisfy $(18.8, 10^{-7})$-DP.

with a batch size of 32 records and a tuned learning rate of 0.005, and we increase the learning rate by $\approx \sqrt{2}$ for every 2x increase in the batch size. For all the experiments with FedAvg, we find that using a constant learning rate provides the best utility across the different batch sizes, and thus, we keep it fixed.

Opt.	Data	Batch Size	RS, BS	Acc. (%), Perp.
SGD	IID	32 rec.	49, 43	23.9, 63
		64 rec.	51, 39	23.5, 65.1
		128 rec.	46, 36	23.2, 67.6
		256 rec.	46, 32	23, 69.7
	Non-IID	32 rec.	40, 29	23.7, 64.2
		64 rec.	38, 32	23.6, 65.6
		128 rec.	35, 26	23.2, 68.2
		256 rec.	40, 31	23, 69.8
Fed-Avg	IID	500 users	66, 56	24.6, 57.5
		1k users	27, 18	24.5, 58
		2k users	28, 18	24.4, 59.3
		5k users	28, 18	24, 62.5
	Non-IID	500 users	21, 0	24.4, 58.8
		1k users	10, 0	24, 61.2
		2k users	0, 0	24, 61.9
		5k users	0, 0	23.3, 67.9

Table 7: Results for the number of inserted canaries (out of 90) memorized via the RS and BS methods, and utility metrics for various models evaluated at ≈10 epochs of training.

For the models trained with SGD, for both IID and non-IID data we observe that unintended memorization remains comparable for models trained with different batch sizes. However, we see a decrease in the utility as the batch size increases. The decrease in utility is observed for models trained using FedAvg as well, but we also observe a significant drop in the such memorization when training is performed with at least 1000 users per round. Moreover, once the training involves at least 2000 users on non-IID data, both the RS and BS methods are unsuccessful in classifying any of the 90 inserted canaries as memorized.