Decepticons: Corrupted Transformers Breach Privacy in Federated Learning for Language Models

Liam Fowl[*], **Jonas Geiping**[*]
University of Maryland
`{lfowl, jgeiping}@umd.edu`

Steven Reich
University of Maryland

Yuxin Wen
University of Maryland

Wojtek Czaja
University of Maryland

Micah Goldblum
New York University

Tom Goldstein
University of Maryland
`tomg@umd.edu`

Abstract

Privacy is a central tenet of Federated learning (FL), in which a central server trains models without centralizing user data. However, gradient updates used in FL can leak user information. While the most industrial uses of FL are for text applications (e.g. keystroke prediction), the majority of attacks on user privacy in FL have focused on simple image classifiers and threat models that assume honest execution of the FL protocol from the server. We propose a novel attack that reveals private user text by deploying malicious parameter vectors, and which succeeds even with mini-batches, multiple users, and long sequences. Unlike previous attacks on FL, the attack exploits characteristics of both the Transformer architecture and the token embedding, separately extracting tokens and positional embeddings to retrieve high-fidelity text.

1 Introduction

Federated learning (FL) has recently emerged as a central paradigm for decentralized training. Where previously, training data had to be collected and accumulated on a central server, the data can now be kept locally and only model updates, such as parameter gradients, are shared and aggregated by a central party. The central tenet of federated learning is that these protocols enable privacy for users [McMahan and Ramage, 2017, Google Research, 2019]. However, in reality, these federated learning protocols walk a tightrope between actual privacy and the appearance of privacy. Attacks that invert model updates sent by users can recover private information in several scenarios Phong et al. [2017], Wang et al. [2018], Zhu et al. [2019], Geiping et al. [2020], Yin et al. [2021].

Most of the work on gradient inversion attacks so far has focused on image classification problems. Conversely, the most successful industrial applications of federated learning have been in language tasks. There, federated learning is not just a promising idea, it has been deployed to consumers in production, for example to improve keystroke prediction [Hard et al., 2019, Ramaswamy et al., 2019] and settings search on the Google Pixel [Bonawitz et al., 2019]. However, attacks against this area have so far succeeded only on limited examples of sequences with *few* (< 25) tokens [Deng et al., 2021, Zhu et al., 2019, Dimitrov et al., 2022]. This leaves the impression that these models are difficult to attack, and limited aggregation is sufficient to protect user privacy, without the necessity to employ stronger defenses such as local or distributed differential privacy [Dwork and Roth, 2013].

In this work, we re-investigate the privacy of these applications for transformer models. We focus on the realistic threat model where the server-side behavior is untrusted by the user, and show that a malicious update sent by the server can completely corrupt the behavior of user-side models, coercing them to spill significant amounts of user data. The server can then collect the original words and

[*]Authors contributed equally. Order chosen randomly.

sentences used by the user with straightforward statistical evaluations and assignment problems. We show for the first time that recovery of all tokens and most of their absolute positions is feasible even on the order of *several thousand* tokens and even when applied to small models only 10% the size of BERT discussed for FL use in Wang et al. [2021]. Furthermore, compared to previous work which only discuss attacks for updates aggregated over few users, this attack breaks privacy even when updates are aggregated over more than 100 users.

2 Motivation and Threat Model

At first glance, updates from Transformer architectures might not appear to leak significant amounts of user data. Gradients are naturally averaged over the entire length of the sequence which is usually significant (e.g. 512 tokens). This mixing of information reduces the utility of their content to an attacker. If one were to draw intuition from vision-based attacks, gradients whose components are averaged over 512 images are impossible to invert even for state-of-the-art attacks [Yin et al., 2021].

On the other hand, the task of language modelling appears much more constrained than recovery in vision settings. The attacker knows from the beginning that only tokens that exist in the vocabulary are possible solutions and it is only necessary to find their location from a limited list of known positions and identify such tokens to reconstruct the input sequence perfectly.

Previous gradient inversion attacks in FL have been described in Deng et al. [2021], Zhu et al. [2019], Dimitrov et al. [2022] and have focused on optimization-based reconstruction in the *honest-but-curious* server model. Our work is further related to work that investigates the unintended memorization abilities of fully trained, but benign, models as described in Carlini et al. [2021], Inan et al. [2021], Brown et al. [2021].

Threat Model We consider the threat model of an untrusted server that is interested in recovering private user data from user updates in FL. Both user and server are bound by secure implementations to follow a known federated learning protocol, and the model architecture is compiled into a fixed state and verified by the same implementation to be a standard Transformer architecture. The user downloads the server state and returns their model updates according to protocol.

This threat model is the most natural setting from a user-centric perspective. Stronger threat models would, for example, allow the server to execute arbitrary code on user devices, but such threats are solvable by software solutions such as secure sandboxing [Frey, 2021].

However, such precautions do not stop an *adversarial server* from sending malicious updates [Wang et al., 2021]. Such attacks are naturally ephemeral - the server can send benign server states nearly all of the time to all users, then switch to a malicious update for single round and group of users (or single secure aggregator), collect user information, and return to benign updates immediately after. Such an attack can be launched by any actor with temporary access to the server states sent to users, including, aside from the server owner, temporary breaches of server infrastructure, MITM attacks or single disgruntled employees.We argue that server states sent to users, especially for very expressive models such as transformers, should be treated by user applications as analogue to *untrusted code* that operates on user data and handled with the appropriate care.

3 How to Program Malicious Transformer Parameters

In this section we will describe this strategy which is comprised of multiple parts. We first describe how the attacker recovers a bag-of-words of the user tokens, and then show how the attacker recovers the position of each token for a single sequence of tokens. Finally, we extend this to multiple sequences, i.e. aggregations of gradient updates from multiple users or multiple sequences from a single user.

3.1 Recovering Token Frequencies

An attacker can immediately retrieve the bag-of-words (the unordered list of all tokens and their assorted frequencies) of the user data from the gradient of the token embedding. Melis et al. [2019] previously identified that unique tokens can be recovered due to the sparsity of rows in the token embedding gradient, as visualized in the left plot of Appendix Figure 4. But, perhaps surprisingly,

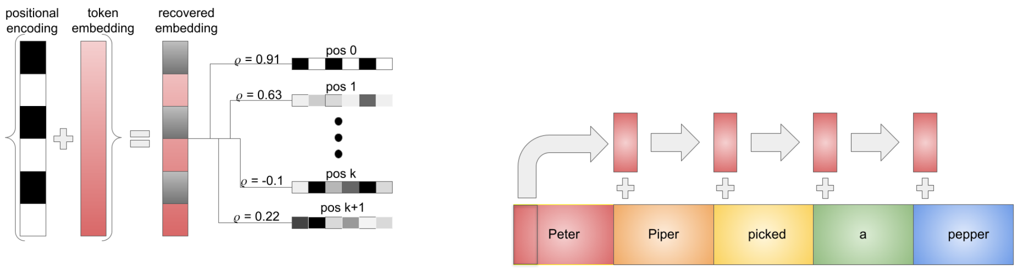


Figure 1: **Left:** A high-level schematic for position assignment. We find correlations ρ between known positions and recovered embeddings, and solve a linear sum assignment problem to determine which position is most likely for the recovered embedding. **Right:** A high-level schematic of MHA manipulation. The MHA block attends to the first word identically for every input sequence, encoding a part of the first token for each embedding in the entire sequence.

we find that even the frequency of all words can be extracted.This frequency estimation can either be triggered by analyzing either the bias gradient of the decoding (last linear) layer, or the norm of the embedding matrix. In both cases, the magnitude of the update of each row of a random embedding matrix is proportional to the frequency of word usage, allowing for a greedy estimation by adapting the strategy of Wainakh et al. [2021] - originally proposed to extract labels in classification tasks.

This first insight already appears to have been overlooked in optimization-based attacks [Deng et al., 2021, Dimitrov et al., 2022] and does not require any malicious modifications.

3.2 Recovering Token Positions in a Sequence

Even for a single sequence of tokens, the number of possible orderings of leaked tokens means that for an attacker, breaching user privacy with information only about leaked tokens becomes intractable. However, once token embeddings are fed into a Transformer-based model, positional information is added to each token - either through a learned positional encoding, or a fixed one as in the original Transformer architecture. An attack that can read out the position of each token can rearrange the bag-of-words from Section 3.1 into the originally private sequence of user tokens.

To this end, we leverage the numerous large linear layers found in the feed-forward blocks of a Transformer-based model, combined with recent strategies for separating gradient signals [Fowl et al., 2021]. In essence, this strategy encodes a measurement function into each weight row of a linear layer and an increasing offset into each entry of the bias vector - allowing individual embeddings to be extracted from gradient information (see Appendix B.2, Figure 13 for more details).

However, this is only part of the battle (for the attacker), as the recovered embeddings themselves correspond to (layer-normalized) sums of token and positional embeddings and have to be associated with a single token id and position. We show that an attacker can accomplish this by first solving a linear sum assignment problem on the recovered embeddings and known positional embeddings E_{pos}. As seen in Figure 1, the attacker can calculate the correlation between the recovered embeddings and the positional embedding, as well as known token embeddings, and solve a linear sum assignment problem to retrieve both a set of tokens, and their corresponding positions (see Appendix and Alg. 1)

3.3 Disambiguating Multiple Sequences

Recovering multiple sequences - either from user data comprising multiple separate sentences of tokens, or aggregates of multiple users - presents the most difficult task for the attacker. Naively using the strategy from Section 3.2 can only recover a partially ordered set of tokens. For example, if a model update consists of five user sequences, then the attacker recovers five first tokens, five second tokens, and so on. However, grouping these tokens into salient sequences quickly becomes intractable as the number of possible groupings grows as n^l for n users sending updates on sequences of length l. Further complicating matters for the attacker is that no learned parameters (and thus no parameters returning gradients) operate on the entire length of a sequence in the Transformer model.

With this as motivation, we unearth a mechanism by which an attacker can disambiguate user sequences, even when model updates are aggregated over a large number of separate sequences. The attacker can do this by modifying W_Q, W_K, W_V - the query, key, and value weights of the attention mechanism to encode unique information about each sequence into a predefined location in the embedding of each token in the sequence. Then, we can read out a unique quantity for each sequence and thus assign tokens to separate sequences. This behavior is illustrated in Figure 1 (b), and further described in Appendix B.3.

3.4 Putting It All Together

We find that an attacker can combine the previous three mechanisms to extract a frightening amount of user data from a federated update. A more detailed description of the synthesis of these mechanisms can be found in Appendix B.4

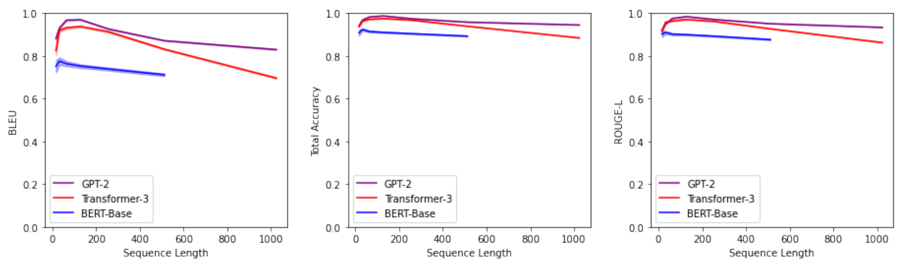


Figure 2: Baseline results for our method for different popular architectures and metrics for variable length sequence input (batch size 1). Note BERT's positional encoding caps out at sequence length 512, so we end our experiments for BERT there.

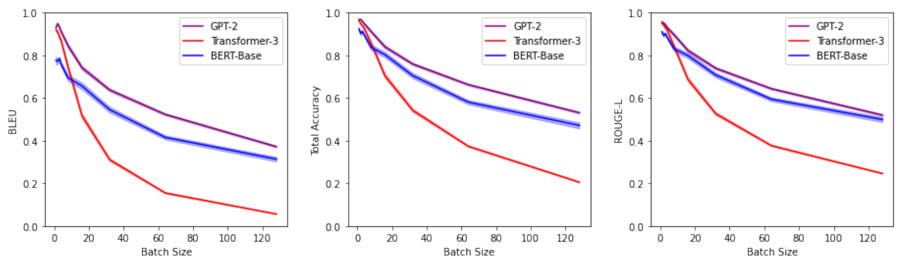


Figure 3: Results for our method for different popular architectures and metrics for variable length batch size (sequence length 32).

For evaluation, we consider three model architectures of differing sizes: first, the small 3-layer Transformer architecture discussed as a template for Transformers in federated learning scenarios in Wang et al. [2021] (11 million parameters), second, the BERT-Base model [Devlin et al., 2019] also attacked in previous work [Deng et al., 2021, Zhu et al., 2019] (110 million parameters), and finally, the smallest GPT-2 variation [Radford et al., 2019] (124 million parameters). We train the small Transformer and GPT-2 as causal language models and BERT as masked language model.

Single Sequence. As a baseline, we begin experimenting with single sequence recovery - a problem that has proven difficult for previous attacks on text using even moderately sized sequences. This corresponds to the scenario wherein a single user's update on a single sequence is received. In Figure 2, we find that even for sequences with > 1000 tokens, for large models like GPT-2, we recover $> 90\%$ of the ground-truth tokens in their *exact* position.

Multi Sequence. In addition to reconstructing single sequences, as discussed in Section 3.4, our multi-head attention strategy allows an attacker to reconstruct multiple sequences. This applies to an update from a single user with more than one sequence, and also multiple users aggregating model updates. To the best of our knowledge, we are the first attack to evaluate and explicitly address the problem of multi-user updates. We find that for a sequence length of 32, the proposed attack can recover almost all tokens used in the user updates, and $> 50\%$ of tokens at their *exact* position for > 100 users. Further experimentation for longer sequences can be found in the appendix.

We provide additional qualitative results in Appendix H, as well as a comparison to the leading honest-but-curious attack, where we find that our method far outperforms the current art.

Acknowledgments

This work is supported by ONR MURI program, DARPA GARD (HR00112020007), and the National Science Foundation (IIS-2212182 and DMS-1912866). Further support was provided by Capital One Bank.

References

Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential Privacy Has Disparate Impact on Model Accuracy. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15479–15488. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9681-differential-privacy-has-disparate-impact-on-model-accuracy.pdf`.

Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection Against Reconstruction and Its Applications in Private Federated Learning. *arXiv:1812.00984 [cs, stat]*, June 2019. URL `http://arxiv.org/abs/1812.00984`.

Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the Curious Abandon Honesty: Federated Learning Is Not Private. *arXiv:2112.02918 [cs]*, December 2021. URL `http://arxiv.org/abs/2112.02918`.

Kallista Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards Federated Learning at Scale: System Design. *arXiv:1902.01046 [cs, stat]*, March 2019. URL `http://arxiv.org/abs/1902.01046`.

Paul S. Bradley, Kristin P. Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, June 2021. doi: 10.1145/3406325.3451131. URL `http://arxiv.org/abs/2012.06421`.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings. *arxiv:1812.01097[cs, stat]*, December 2019. doi: 10.48550/arXiv.1812.01097. URL `http://arxiv.org/abs/1812.01097`.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting Training Data from Large Language Models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021. ISBN 978-1-939133-24-3. URL `https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting`.

David F. Crouse. On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1679–1696, August 2016. ISSN 1557-9603. doi: 10.1109/TAES.2016.140952.

Jieren Deng, Yijue Wang, Ji Li, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. TAG: Gradient Attack on Transformer-based Language Models. *arXiv:2103.06819 [cs]*, September 2021. URL `http://arxiv.org/abs/2103.06819`.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual Energy-Based Models for Text Generation. *arXiv:2004.11714 [cs]*, April 2020. URL `http://arxiv.org/abs/2004.11714`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL `http://arxiv.org/abs/1810.04805`.

Dimitrios Dimitriadis, Mirian Hipolito Garcia, Daniel Madrigal Diaz, Andre Manoel, and Robert Sim. FLUTE: A Scalable, Extensible Framework for High-Performance Federated Learning Simulations. *arxiv:2203.13789[cs]*, March 2022. URL http://arxiv.org/abs/2203.13789.

Dimitar I. Dimitrov, Mislav Balunović, Nikola Jovanović, and Martin Vechev. LAMP: Extracting Text from Gradients with Language Model Priors. February 2022. URL https://arxiv.org/abs/2202.08827v1.

Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013. ISSN 1551-305X, 1551-3068. doi: 10.1561/0400000042. URL http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042.

Liam H. Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models. In *International Conference on Learning Representations*, September 2021. URL https://openreview.net/forum?id=fwzUgo0FM9v.

Suzanne Frey. Introducing Android's Private Compute Services, September 2021. URL https://security.googleblog.com/2021/09/introducing-androids-private-compute.html.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients - How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, December 2020. URL https://proceedings.neurips.cc//paper_files/paper/2020/hash/c4ede56bbd98819ae6112b20ac6bf145-Abstract.html.

Team Google Research. Federated Learning, May 2019. URL https://federated.withgoogle.com.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated Learning for Mobile Keyboard Prediction. *arXiv:1811.03604 [cs]*, February 2019. URL http://arxiv.org/abs/1811.03604.

Florian Hartmann. Predicting Text Selections with Federated Learning, November 2021. URL http://ai.googleblog.com/2021/11/predicting-text-selections-with.html.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Huseyin A. Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. Training Data Leakage Analysis in Language Models. *arxiv:2101.05405[cs]*, February 2021. doi: 10.48550/arXiv.2101.05405. URL http://arxiv.org/abs/2101.05405.

Bargav Jayaraman and David Evans. Evaluating Differentially Private Machine Learning in Practice. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 1895–1912, 2019. ISBN 978-1-939133-06-9. URL https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman.

R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, December 1987. ISSN 1436-5057. doi: 10.1007/BF02278710. URL https://doi.org/10.1007/BF02278710.

Peter Kairouz, Ziyu Liu, and Thomas Steinke. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5201–5212. PMLR, July 2021a. URL https://proceedings.mlr.press/v139/kairouz21a.html.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, 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. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]*, March 2021b. URL `http://arxiv.org/abs/1912.04977`.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What Can We Learn Privately? *SIAM Journal on Computing*, 40(3):793–826, January 2011. ISSN 0097-5397. doi: 10.1137/090756090. URL `https://epubs.siam.org/doi/10.1137/090756090`.

Wenqi Li, Fausto Milletarì, Daguang Xu, Nicola Rieke, Jonny Hancox, Wentao Zhu, Maximilian Baust, Yan Cheng, Sébastien Ourselin, M. Jorge Cardoso, and Andrew Feng. Privacy-Preserving Federated Brain Tumour Segmentation. In Heung-Il Suk, Mingxia Liu, Pingkun Yan, and Chunfeng Lian, editors, *Machine Learning in Medical Imaging*, Lecture Notes in Computer Science, pages 133–141, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32692-0. doi: 10.1007/978-3-030-32692-0_16.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large Language Models Can Be Strong Differentially Private Learners. In *International Conference on Learning Representations*, March 2022. URL `https://openreview.net/forum?id=bVuP3ltATMz`.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

Brendan McMahan and Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data, April 2017. URL `http://ai.googleblog.com/2017/04/federated-learning-collaborative.html`.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning Differentially Private Recurrent Language Models. *arXiv:1710.06963 [cs]*, February 2018. URL `http://arxiv.org/abs/1710.06963`.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 691–706, May 2019. doi: 10.1109/SP.2019.00029.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. November 2016. URL `https://openreview.net/forum?id=Byj72udxe`.

Maxence Noble, Aurélien Bellet, and Aymeric Dieuleveut. Differentially Private Federated Learning on Heterogeneous Data. *arxiv:2111.09278[cs, math, stat]*, February 2022. URL `http://arxiv.org/abs/2111.09278`.

Documentation ONNX. ONNX Operator Schemas. Open Neural Network Exchange, January 2022. URL `https://github.com/onnx/onnx/blob/50a1981dc3d50af13075cec33b08b4c87fb0e41f/docs/Operators.md`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://aclanthology.org/P02-1040`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*, Long Beach, CA, 2017. URL `https://openreview.net/forum?id=BJJsrmfCZ`.

Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, Sudeep Agarwal, Julien Freudiger, Andrew Byde, Abhishek Bhowmick, Gaurav Kapoor, Si Beaumont, Áine Cahill, Dominic Hughes, Omid Javidbakht, Fei Dong, Rehan Rishi, and Stanley Hung. Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications. *arXiv.2102.08503*, February 2021. doi: 10.48550/arXiv.2102.08503. URL `https://arxiv.org/abs/2102.08503v1`.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. Technical Report 715, 2017. URL `http://eprint.iacr.org/2017/715`.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI*, page 24, 2019.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated Learning for Emoji Prediction in a Mobile Keyboard. *arXiv:1906.04329 [cs]*, June 2019. URL `http://arxiv.org/abs/1906.04329`.

Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

Jonathan Ullman. Tight Lower Bounds for Locally Differentially Private Selection. *arXiv:1802.02638 [cs]*, May 2021. URL `http://arxiv.org/abs/1802.02638`.

Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User Label Leakage from Gradients in Federated Learning. *arXiv:2105.09369 [cs]*, June 2021. URL `http://arxiv.org/abs/2105.09369`.

Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H. Brendan McMahan, Blaise Aguera y Arcas, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, Suhas Diggavi, Hubert Eichner, Advait Gadhikar, Zachary Garrett, Antonious M. Girgis, Filip Hanzely, Andrew Hard, Chaoyang He, Samuel Horvath, Zhouyuan Huo, Alex Ingerman, Martin Jaggi, Tara Javidi, Peter Kairouz, Satyen Kale, Sai Praneeth Karimireddy, Jakub Konecny, Sanmi Koyejo, Tian Li, Luyang Liu, Mehryar Mohri, Hang Qi, Sashank J. Reddi, Peter Richtarik, Karan Singhal, Virginia Smith, Mahdi Soltanolkotabi, Weikang Song, Ananda Theertha Suresh, Sebastian U. Stich, Ameet Talwalkar, Hongyi Wang, Blake Woodworth, Shanshan Wu, Felix X. Yu, Honglin Yuan, Manzil Zaheer, Mi Zhang, Tong Zhang, Chunxiang Zheng, Chen Zhu, and Wennan Zhu. A Field Guide to Federated Optimization. *arXiv:2107.06917 [cs]*, July 2021. URL `http://arxiv.org/abs/2107.06917`.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. *arXiv:1812.00535 [cs]*, December 2018. URL `http://arxiv.org/abs/1812.00535`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*, July 2020. URL `http://arxiv.org/abs/1910.03771`.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See Through Gradients: Image Batch Recovery via GradInversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021. URL `https://openaccess.thecvf.com/content/CVPR2021/html/Yin_See_Through_Gradients_Image_Batch_Recovery_via_GradInversion_CVPR_2021_paper.html`.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 14774–14784. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9617-deep-leakage-from-gradients.pdf`.

A X-Risk Sheet

Individual question responses do not decisively imply relevance or irrelevance to existential risk reduction. Do not check a box if it is not applicable.

A.1 Long-Term Impact on Advanced AI Systems

Here we describe how our work reveals a significant privacy risk associated with AI and helps lead the community toward a safer federated learning regime.

1. **Overview.** How is this work intended to reduce existential risks from advanced AI systems?
   Answer: Data privacy is of vital importance both for users' notions of safety, but also for the protection of sensitive information. If federated learning pipelines are deemed safe by large tech companies, they could be deployed in several security-critical areas including email, messaging, etc. which could be leveraged by malicious actors to extract private and important information.
2. **Direct Effects.** If this work directly reduces existential risks, what are the main hazards, vulnerabilities, or failure modes that it directly affects?
   Answer: We believe that knowledge of attacks like the one we unearth is necessary for counteracting these attacks in the future. We do not directly reduce the risk in this work, but we believe proliferating knowledge of these types of attacks is the first piece of the puzzle.
3. **Diffuse Effects.** If this work reduces existential risks indirectly or diffusely, what are the main contributing factors that it affects?
   Answer: We hope the publication of this work will spur community action to develop coherent and secure procedures for federated learning systems which include security measures for malicious/untrusted servers.
4. **What's at Stake?** What is a future scenario in which this research direction could prevent the sudden, large-scale loss of life? If not applicable, what is a future scenario in which this research direction be highly beneficial?
   Answer: While federated learning is currently used for seemingly benign applications like smart-reply, as federated learning becomes more mature, and profitable, it is likely that it will be deployed in more settings, and be made available to third parties to leverage large user bases. In the worst-case scenario, an attack like ours could be deployed by malicious actors, including nation-states, interacting with user data via third-party APIs made available by tech companies. This could mean that any sensitive user information could be compromised - including information like secure passwords exchanged over messages.

   The types of data stealing we demonstrate are possible could be used for mass surveillance and therefore for a catastrophic loss of rights or oppression, for example by governments. Federated learning is considered the standard for user-cooperative training of large-scale model, and in fact has already seen industrial applications in different settings. With the proliferation of this framework comes immense risk. Revealing the capability of servers to breach user privacy and pushing the community to develop advanced protection algorithms could prevent such catastrophes.
5. **Result Fragility.** Do the findings rest on strong theoretical assumptions; are they not demonstrated using leading-edge tasks or models; or are the findings highly sensitive to hyperparameters? **Answer** We do not rely on any strong theoretical qassumptions, and we demonstrate results. inpractical scenarios including attacking architectures which are known to be used in federated settings.
6. **Problem Difficulty.** Is it implausible that any practical system could ever markedly outperform humans at this task? ☐
7. **Human Unreliability.** Does this approach strongly depend on handcrafted features, expert supervision, or human reliability? ☐
8. **Competitive Pressures.** Does work towards this approach strongly trade off against raw intelligence, other general capabilities, or economic utility? ☐

A.2 Safety-Capabilities Balance

In this section, please analyze how this work relates to general capabilities and how it affects the balance between safety and hazards from general capabilities.

1. **Overview.** How does this improve safety more than it improves general capabilities?
   Answer: We demonstrate that safety mechanisms like secure aggregation, which were previously thought to be sufficient to ensure user privacy, are in fact lacking in the face of an attack like the one we propose. Because of this, we advocate for stronger, differential-privacy based safety mechanisms for federated learning.
2. **Red Teaming.** What is a way in which this hastens general capabilities or the onset of x-risks?
   Answer: While our paper describes an attack, not a defense, we believe that knowledge is power, and attacks like ours being out in the open is a better way to deal with these risks as it spurs future defensive developments.
3. **General Tasks.** Does this work advance progress on tasks that have been previously considered the subject of usual capabilities research? ☒
4. **General Goals.** Does this improve or facilitate research towards general prediction, classification, state estimation, efficiency, scalability, generation, data compression, executing clear instructions, helpfulness, informativeness, reasoning, planning, researching, optimization, (self-)supervised learning, sequential decision making, recursive self-improvement, open-ended goals, models accessing the Internet, or similar capabilities? ☐
5. **Correlation With General Aptitude.** Is the analyzed capability known to be highly predicted by general cognitive ability or educational attainment? ☐
6. **Safety via Capabilities.** Does this advance safety along with, or as a consequence of, advancing other capabilities or the study of AI? ☒

Algorithm 1 Decepticon Data Readout

```
1:  Input: Transformer model $T$, malicious server state $\theta_0$, user update $\theta_1$, number of expected
    sequences $n$.
2:  $E_{\text{unordered}}$ = Embeddings of estimated bag-of-words of leaked tokens
3:  $E_{\text{breached}}$ = Breached Embeddings $\frac{\nabla_{W_i^j} - \nabla_{W_{i+1}^j}}{\nabla_{b_i^j} - \nabla_{b_{i+1}^j}}$ for linear layers $j = 1, ..L$ and rows $i = 1, ..., r$.
4:  $L_{\text{batch}}$ = Batch label for each $E_{\text{breached}}[: v]$ from clustering into $n$ clusters of maximal size $l$ via
    constrained k-means.
5:  $E_{\text{positions}} \leftarrow$ Known positional embeddings
6:  for b in $0...B$ do
7:      $E^b_{\text{ordered}}$ = Match($E_{\text{positions}}[v :]$, $E_{\text{breached}}[L_{\text{batch}} = b, v :]$)
8:  end for
9:  $E_{\text{ordered}}$ = concatenate $\{E^b_{\text{ordered}}\}_{b=1}^B$
10: while empty rows in $E_{\text{ordered}}$ do
11:     $L_{\text{free positions}}$ = indices of empty rows in $E_{\text{ordered}}$
12:     $E_{\text{ordered}}$ =Match($E_{\text{positions}}[L_{\text{free positions}}]$, $E_{\text{breached}}[: v]$
13: end while
14: $E_{\text{ordered, no pos.}}$ = $E_{\text{ordered}}$ - $E_{\text{positions}}$
15: $T_{\text{final tokens}}$ = Indices of Match($E_{\text{ordered, no pos.}}$, $E_{\text{unordered}}$)
```

B Technical Details

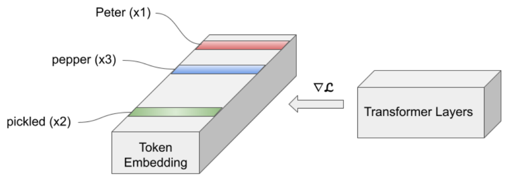
We implement all attacks in an extensible `PyTorch` framework [Paszke et al., 2017] for this type of attack which allows for a full reproduction of the attack and which we attach to this submission. We utilize `huggingface` models and data extensions for the text data [Wolf et al., 2020].The attack is separated into two parts. The first part is the malicious modification of the server parameters, which is triggered immediately before the server sends out their model update payload to users.

We implement the malicious parameter modifications as described in Sec. 3. We initialize from a random model initialization and reserve the first 6 entries of the embedding layer for the sentence encoding for the Transformer-3 model and the first 32 entries for the larger BERT and GPT-2 models. The corresponding entries in the positional and token embeddings are reset to 0. In the MHA modification, we choose a softmax skewing of $1e8$, although this value can be significantly lower in practice as well. We reserve the last entry of the embedding layer for gradient flow to each linear layer, so that the attack is able to utilize all layers as described. This entry is scaled by $\varepsilon = 1e - 6$ for the smaller Transformer-3 model and $\varepsilon = 1e - 8$ for the larger models, so that the layer normalization is not skewed by large values arising from the last embedding entry.

For the attack part, we retrieve the gradient update on the server and run all recovery computations in single float precision. We first run a token recovery as discussed in Sec 3.1, which we additionally summarize in Sec. 3.4 and then proceed with the attack following Algorithm 1. For the sentence labeling we cluster using constrained K-means as described in Bradley et al. [2000]. For all assignment problems we utilize the linear sum assignment solver proposed in Crouse [2016] which is a modification of the shortest augmenting path strategy originally proposed in Jonker and Volgenant [1987].

During the quantitative evaluation, we trial 100 users each with a request for updates of size sequence length $\times$ batches. We skip all users which do *not* own enough data and do not pad user data with [PAD] tokens, which we think would skew results as it would include a large number of easy tokens to recover. All measurements are hence done on non-trivial sentences, which are concatenated to reach the desired sequence length. Each user's data is completely separated, representing different wikipedia articles as described in the main body. Overall, the attack is highly successful over a range of users even with very different article content.

To compute metrics, we first resort the recovered batches toward the ground truth order by assigning to ground truth batches based on total token accuracy per sentence. This leads to a potential underestimation of the true accuracy and ROUGE metrics of the recovery as sentence are potentially mismatched. We compute Rouge [Lin, 2004] scores based on the sorted batches and BLEU scores [Papineni et al., 2002] (with the default `huggingface` implementation) by giving all batches as



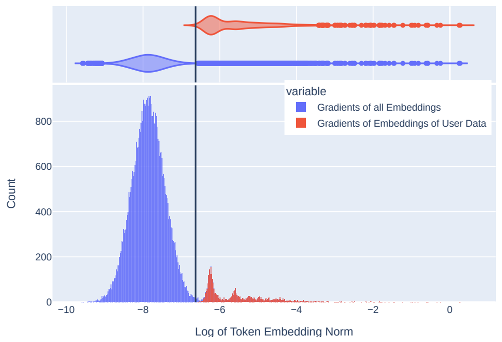


Figure 4: **Left:** A high-level schematic of a simple case of token leaking. The token embedding layer can leak tokens and frequency solely through its sparse gradient entries. **Right:** Distribution and Histogram of log of norms of all token embedding gradients for GPT-2 for 13824 tokens. In this case, gradient entries are non-sparse due to the tied encoder-decoder structure of GPT, but the embeddings of true user data (red) are clearly separable from the mass of all embeddings (blue) by a simple cutoff.

references. We measure total accuracy as the number of tokens that are perfectly identified and in the correct position, assigning no partial credit for either token identity or position. This is notably in contrast to the naming of metrics in Deng et al. [2021] where accuracy refers to only overall token accuracy. We refer to that metric as "token accuracy", measuring only the overlap between reference and reconstruction sentence bag-of-words, but report only the total accuracy, given that token accuracy is already near-perfect after the attack on the bag-of-words described in Sec 3.1.

B.1 Token extraction

Here, we provide a schematic for our token extraction method, which works even in the presence of models using tied embeddings. See Figure 4 and Algorithms 2, 3 for more details.

B.2 Details on embedding extraction

To go further into detail into this modification of Fowl et al. [2021], we denote a forward pass on the first layer (of the feed-forward component) of the i^{th} Transformer block by $\phi_1^i = W_{1,i}x + b_{1,i}$. It is straightforward to derive that for a single embedding vector x passed through ϕ_1^i, for the j^{th} row of the linear layer, $W_{1,i}^j$, as long as $\frac{\partial \mathcal{L}}{\partial b_{1,i}^j} \neq 0$, then $\nabla_{W_{1,i}} \mathcal{L} / \frac{\partial \mathcal{L}}{\partial b_{1,i}^j} = x$. Simply put, linear layers encode their inputs directly in the gradient information of their weights and biases.

We sample a Gaussian vector $m \sim \mathcal{N}(\vec{0}, \mathbb{1}_d)$ where $d = d_{model}$, and identically set each row j of the weights of $\phi_{1,j}^i = m$, and the biases of this layer we set to $b_1^i = [c_{i \cdot k}, \ldots, c_{(i+1) \cdot k}]$. Here, k is the width each linear layer, and $c_j = -\Phi^{-1}(\frac{j}{M})$ for Φ^{-1}, the inverse of the standard Gaussian CDF, and M is the sum over the widths of all included Transformer blocks Note that we can well estimate the rough mean and variance of this quantity (needed to normalize it) from small volume of public text (see the Shakespeare dataset in the Appendix). Then, when $\langle x, m \rangle$ uniquely falls between two values $[c_{j-1}, c_j]$, the gradient of the corresponding linear layer with those biases contains unique information that can be inverted to recover the embedding x. The attacker can set the second linear layer in the feed-forward portion of a Transformer block to collapse the outputs of the first linear layer into a small quantity to be added with the residual connection to a single entry for the original embeddings, thus enabling non-zero gradients in each linear layer, while also not perturbing the embeddings too much. This leverages the use of all feed-forward blocks for this attack. See Appendix Figure 13.

B.3 Details on multi-sequence extraction

Let W_Q, W_K, W_V represent the query, key, and value weight matrices respectively, and let b_Q, b_K, b_V represent their biases. For simplicity of presentation, we explain this attack on a single head, but it is easily adapted to multi-head attention. We first set the W_K matrix to the identity ($\mathbb{1}_{d_{model}}$), and $b_K = \vec{0}$. This leaves incoming embeddings unaltered. Then, we set $W_Q = \vec{0}$, and $b_Q = \gamma \cdot \vec{p}_0$ where $\vec{p}_0$ is the first positional encoding. Here we choose the first position vector for simplicity, but there are many potential choices for the identifying vector. This query matrix then transforms each embedding identically to be a scaled version of the first positional encoding. We then set $W_V = \mathbb{1}_{d'}$ to be a partial identity (identity in the first d' entries where $d' \leq d$ is a hyperparameter that the server controls). Finally, we set $b_v = \vec{0}$.

Now, we investigate how these changes transform an incoming sequence of embeddings. Let $\{x_i\}_{i=0}^{l-1}$ be embeddings for a sequence of length l that enters the attention mechanism. x_0 is the embedding corresponding to the first token in the sequence, so x_0 is made up of the token embedding for the first token in the sequence, and the first positional encoding. W_K produces keys K that exactly correspond to the incoming embeddings, however, W_Q, b_Q collapses the embeddings to produce Q, consisting of l identical copies of a single vector, the first positional encoding. Then, when the attention weights are calculated as:

$$\operatorname{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right),$$

the attacker finds that the first embedding dominates the attention weights for all the embeddings, as the query vectors all correlate with the first embedding the most. In fact, the γ parameter can effectively turn the attention weights to a delta function on the first position. Finally, when the attention weights are used to combine the values V with the embeddings, by construction, a part of the embedding for the first word in the sequence is identically added to each other word in that sequence. So the embeddings are transformed as $\{x_i\}_{i=0}^{l-1} \to \{x_i + x_{0,d'}\}_{i=0}^{l-1}$ where $x_{0,d'}$ is a vector where the first d' entries are the first d' entries of x_0, and the other $d_{model} - d'$ entries are identically 0.

If the attacker chooses to perform this modification on the first Transformer block, this means that embeddings that the attacker recovers from each of the linear layers (as detailed in Section 3.2) now contain unique information about the sequence from which they came. The attacker can then calculate correlations between the first part of each recovered embedding and cluster these correlations in order to group each embedding into a sequence with other embeddings.

B.4 Details on putting the mechanisms together

Now that we have introduced the three main mechanisms for our proposed attack, we describe the procedure from start to finish and introduce additional mechanisms that are useful for the attack. The attack begins after a user or aggregate group of users has computed their model update based on the corrupted parameters. Then, the server retrieves their update and begins the inversion procedure.

The server first computes normalized un-ordered embeddings based on the bag-of-words from Section 3.1, normalized positional embeddings, based on the known max. sequence length (normalizing by the first layer norm, as the linear layers appear after the layer norm operation). The server then computes the actual “breached" embeddings, E_{breached}, from the bins inserted into the linear layers. All gradient rows in these linear layers record the sum over all embeddings whose inner product with the measurement vector is smaller than the bias of the row c_i. Conversely, by subtracting subsequent rows, the server recovers in each row the embedding with inner product within the range $[c_i, c_{i+1}]$. All rows with non-zero bias now correspond to such bins which contain 1 or more embeddings. Dividing these embeddings by their bias recovers a number of breached embeddings, which correspond directly to the input of the linear layers (although some recoveries may be mixtures of multiple embeddings).

The first d' entries of each embedding encode the sequence identity as described in Section 3.3. The server can hence group the embeddings E_{breached} uniquely into their sequences by a clustering algorithm. We utilize constrained K-Means, as the maximal sequence length and hence cluster size is known as described in [Bradley et al., 2000].

We can now proceed for each sequence separately and match all found embeddings for the sentence to the known positional embeddings $E_{\text{positions}}$, thereby ordering the embeddings E_{breached} for each sequence. Due to mixtures, some positions will not be filled after this initial matching step. We thus iteratively fill up free positions with the best-matching embeddings from E_{breached}, even if they have been used before in other positions. After this procedure all entries in E_{ordered} will be filled with entries from E_{breached}. We can then compute $E_{\text{ordered, no pos.}}$ by subtracting the positional encodings from these ordered embeddings, which should be an estimate of the token embedding at the given position. Finally we match $E_{\text{ordered, no pos.}}$ with the embeddings of tokens recovered in the beginning

$E_{\text{unordered}}$ to recover the most likely identity of each token in the sequence. We provide additional details in the appendix.

In general, this process works very well to recover even several batches of long sequences. In the limit of more and more tokens the quality eventually degrades as more and more breached embeddings are mixtures of multiple inputs, making the identification of their position and token id less certain. This is partially compensated for by the iterative filling of missing positions. When in the regime of a massive number of tokens, only a subset of positions are accurately recovered, and the remaining tokens are inaccurate, appearing in wrong positions in the input. This process is semi-random, so that at even for a massive number of tokens there will still exist some batches which can be recovered very well.

B.5 Evaluation details

Here we describe details on how we evaluate our proposed attack.

We assemble data from *wikitext* [Merity et al., 2016], which we partition into separate users by articles and tokenize using the GPT-2 (BPE) tokenizer for the small Transformer and GPT-2, and the original BERT (WordPiece) tokenizer for BERT. We then evaluate the attack over a range of sequence lengths and batch sizes. For quantitative evaluations, we always report average scores over the first 100 users (i.e articles), which are long enough to fit batch size $\times$ sequence length tokens. We focus on fedSGD, i.e. single gradient updates from all users in this work, but note that related attacks in Fowl et al. [2021] are also applicable in fedAVG scenarios - although, arguably, the server controls the client learning rate and could thus turn all updates into gradient updates by choosing the client learning rate maliciously.

We evaluate all attacks using BLEU score [Papineni et al., 2002], *total* accuracy, and ROUGE-L [Lin, 2004]. Note that our total accuracy score is stronger than the token accuracy as described in previous works, such as Deng et al. [2021], who use this term to describe bag-of-words accuracy (which we already solve in Section 3.1). In constrast, we measure total accuracy and only count success if both token id *and* position are correct.

C Algorithm Details

We detail sub-algorithms as additional material in this section. Algorithm 2 and Algorithm 3 detail the token recovery for transformers with decoder bias and for transformer with a tied embedding. These roughly follow the principles of greedy label recovery strategy proposed in Wainakh et al. [2021] and we reproduce them here for completeness, incorporating additional considerations necessary for token retrieval.

Algorithm 2 Token Recovery (Decoder Bias)

```
 1: Input: Decoder bias gradient $g_b$, embedding gradient $g_e$, sequence length $s$, number of sequences $n$.
 2: $v_{\text{tokens}} \leftarrow$ all indices where $g_b < 0$
 3: $v_e \leftarrow$ all indices where $g_e < 0$
 4: $v_{\text{tokens}}$ append $v_{\text{tokens}} \setminus v_e$
 5: $m_{\text{impact}} = \frac{1}{sn} \sum_{i \in v_{\text{tokens}}} g_{bi}$
 6: $g_b[v_{\text{tokens}}] \leftarrow g_b[v_{\text{tokens}}] - m_{\text{impact}}$
 7: while Length of $v_{\text{tokens}} < sn$ do
 8:     $j = \arg\min_i g_{bi}$
 9:     $g_{bj} \leftarrow g_{bj} - m_{\text{impact}}$
10:     $v_{\text{tokens}}$ append $j$
11: end while
```

Algorithm 3 Token Recovery (Tied encoder Embedding)

```
 1: Input: Embedding weight gradient $g_e$, sequence length $s$, number of expected sequences $n$, cutoff factor $f$
 2: $n_{ej} = ||g_{ej}||_2$ for all $j = 1, ...N$ embedding rows
 3: $\mu, \sigma = \text{Mean}(\log n_e), \text{Std}(\log n_e)$
 4: $c = \mu + f\sigma$
 5: $v_{\text{tokens}} \leftarrow$ all indices where $n_e > c$
 6: $m_{\text{impact}} = \frac{1}{sn} \sum_{i \in v_{\text{tokens}}} n_{ei}$
 7: $n_e[v_{\text{tokens}}] \leftarrow n_e[v_{\text{tokens}}] - m_{\text{impact}}$
 8: while Length of $v_{\text{tokens}} < sn$ do
 9:     $j = \arg\min_{i \in v_{\text{tokens}}} n_{ei}$
10:     $n_{ej} \leftarrow n_{ej} - m_{\text{impact}}$
11:     $v_{\text{tokens}}$ append $j$
12: end while
```

D Measurement Transferability

In order to retrieve positionally encoded features from the first linear layer in the feed-forward part of each Transformer block, we create “bins" which partition the possible outcomes of taking the scalar product of an embedding with a random Gaussian vector. To do this, we estimate the mean and variance of such a measurement, and create bins according to partitions of equal mass for a Gaussian with this mean and variance. A natural question arises: how well can we estimate this quantity? If the server does not have much data from the user distribution, will this present a problem for recovering user data? To this end, we demonstrate that the server can estimate these measurement quantities well using a surrogate corpus of data. In Figure 5, we see that the Gaussian fit using the Wikitext dataset is strikingly similar to the one the server would have found, even on a very different dataset (Shakespeare).

We further note that an ambitious server could also directly use model updates retrieved from users in a first round of federated learning, to directly glean the distribution of these linear layers, given that the lowest bin records the average of all activations of this layer. However, given the ubiquity of public text data, this step appears almost unnecessary - but possibly relevant in specialized text domains or when using Transformers for other data modalities such as tabular data [Somepalli et al., 2021].

For the purposes of this work, however, we do make the assumption that the attacker has access to some public corpus of text (possibly dissimilar) on which to estimate the distribution of these quantities.

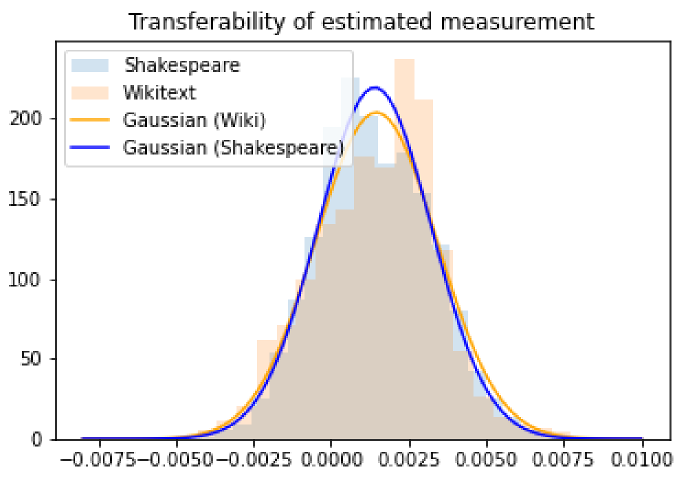


Figure 5: Comparison of Gaussian fit to the measurements of Shakespeare dataset, and a Gaussian fit to the measurements (against the same vector) for the Wikitext dataset.

E Variants and Details

E.1 Masked Language Modelling

For problems with masked language modelling, the loss is sparse over the sequence length, as only masked tokens compute loss. This impacts the strategy proposed in Sec 3.2, as with disabled attention mechanisms in all but the first layer, no gradient information flows to unmasked entries in the sequence. However, this can be quickly solved by reactivating the last attention layer, so that it equally attends to all elements in the sequence with a minor weight increment which we set to 10. This way, gradient flow is re-enabled and all computations can proceed as designed. For BERT, we further disable the default `huggingface` initialiazation of the token embedding, which we reset to a random normal initialization.

Masked language modelling further inhibits the decoder bias strategy discussed in Sec 3.1, as only masked tokens lead to a non-positive decoder bias gradient. However, we can proceed for masked language models by recovering tokens from the embedding layer as discussed for models without decoder bias. The mask tokens extracted from the bias can later be used to fill masked positions in the input.

E.2 GELU

A minor stumbling for the attacker occurs if the pre-defined model uses GELU [Hendrycks and Gimpel, 2016] activation instead of ReLU. This is because GELU does not threshold activations in the same was as ReLU, and transmits gradient signal when the activation < 0. However, a simple workaround for the attacker is to increase the size of the measurement vector and, by doing so, push activations away from 0, and thus more toward standard ReLU behavior. For the main-body experiments, we use ReLU activation for simplicity of presentation, but using the workaround described above, we find in Figure 6, Figure 7 that the performance of the attack against a GELU network is comparable to its level against a ReLU network.

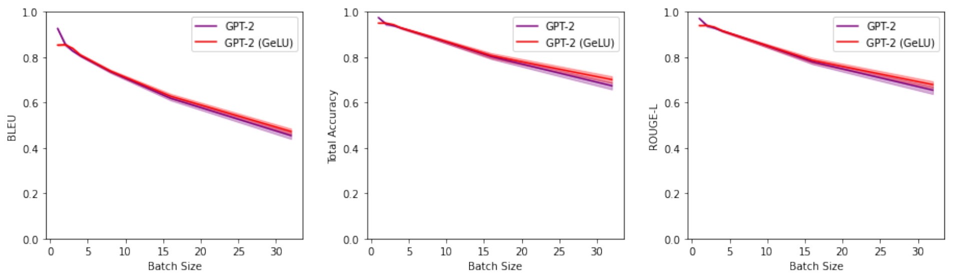


Figure 6: Comparison of GELU and ReLU activation with magnified measurement vector (sequence length 256).

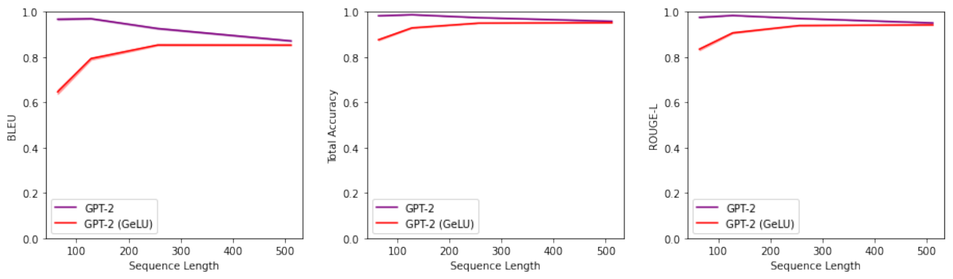


Figure 7: Comparison of GELU and ReLU activation with magnified measurement vector (batch size 1).

E.3 Dropout

All experiments in the main body, including comparisons, have assumed that dropout has been turned off for all models under consideration. In standard applications, dropout can be modified from the server-side, even for compiled and otherwise fixed models [ONNX, 2022]. In our threat model, dropout hence falls into the category of a server-side parameter that a malicious update could turn off. We also investigated the performance of the attack if the attacker cannot alter standard dropout parameters, finding that dropout decreased total accuracy of the attack by about 10%, e.g. from 93.36% for a sequence of length 512 on GPT-2 to 81.25% with dropout.

F Additional Background Material

The attack TAG in Deng et al. [2021] approaches gradient inversion attacks against transformer models from the direct optimization-based angle. This was first suggested in Zhu et al. [2019], who provide some preliminary experiments on recovery of short sequences from BERT. Basically, the attack works to recover user data (x^*, y^*) from the measurement of the gradient on this data g by solving the optimization problem of

$$\min_{x,y} ||\mathcal{L}(n(x,\theta),y) - g||^2, \tag{1}$$

where x and y are the inputs and labels to be recovered, $\mathcal{L}$ is the loss and $n(\cdot,\theta)$ is the language model with parameters θ. This optimization approach does succeed for short sequences, but the optimization becomes quickly stuck in local minima and cannot recover the original input data. Zhu et al. [2019] originally propose the usage of an L-BFGS solver to optimize this problem, but this optimizer can often get stuck in local minima. Deng et al. [2021] instead utilize the "BERT" Adam optimizer with hyperparameters as in BERT training [Devlin et al., 2019]. They further improve the objective by adding an additional term that especially penalizes gradients of the first layers, which we also implement when running their attack. A major problem for the attacks of Zhu et al. [2019] and Deng et al. [2021], however, is the large label space for next-word prediction. In comparison to binary classification tasks as mainly investigated in Deng et al. [2020], the large label space leads to significant uncertainty in the optimization of labels y, which leads their attack to reduce in performance as vocabulary size increases.

We further note that Boenisch et al. [2021] also propose malicious model modifications (as opposed to our more realistic malicious parameter modifications) to breach privacy in FL for text, however the proposed model in their work is a toy two-layer fully connected model that is not a Transformer-based model. In fact, the strategy employed in their attack cannot be deployed against modern language models that do not construct a linear layer of the length of the sequence, aside from handling of facets of modern language models like positional encodings, or attention mechanisms.

G Mitigation Strategies

Several known case studies of FL models deployed in production in Hard et al. [2019], Hartmann [2021] rely only on aggregation to preserve privacy in FL for text. However, the attack we describe in this work shows that aggregation levels considered safe based on previous work may not be enough to protect user privacy in several use cases: The setting deployed to production in Hard et al. [2019] runs FL on users updates with 400 sentences with 4 words per average message sent to the server without further aggregation, well within the range of the investigated attack if trained with a transformer model. We thus briefly discuss other mitigation strategies for attacks like this. We roughly group them into two categories: parameter inspection and differential privacy based defenses.

Parameter inspection and verification of the server state is currently not implemented in any major industrial FL framework [Paulik et al., 2021, Dimitriadis et al., 2022, Li et al., 2019, Bonawitz et al., 2019], but after the publication of this attack, a rule could be designed to mitigate it (which we would encourage!). However, we caution that the design space for attacks as described here seems too large to be defended by inspecting parameters for a list of known attacks.

Based on these concerns, general differential privacy (DP) based defenses continue to be the more promising route. Local or distributed differential privacy, controlled directly by the user and added directly on-device (instead of at the server level as in McMahan et al. [2018]) allows for general protection for users without the need to trust the update sent by the server [Kairouz et al., 2021a, Bhowmick et al., 2019]. Local DP does come at a cost in utility, observed in theoretical [Kairouz et al., 2021b, Kasiviswanathan et al., 2011, Ullman, 2021] as well as practical investigations [Jayaraman and Evans, 2019] when a model is pretrained from scratch [Li et al., 2022, Noble et al., 2022]. DP currently also comes at some cost in fairness, as model utility is reduced most for underrepresented groups [Bagdasaryan et al., 2019]. Yet, as any attack on privacy will break down when faced with sufficient differential privacy, with this work we advocate for strong differential privacy on the user side, as incorporated for example into recent system designs in Paulik et al. [2021].

A central message of our paper is that aggregation alone may not be sufficient to defend against malicious parameter attacks in FL. Further defenses may be needed. Such defenses could include parameter inspection wherein a defender could look for identifiable signatures in the shared parameter vector. For example, the given attack duplicates a measurement vector, leading to low rank linear layers. To the best of our knowledge, there do not exist systematic parameter inspection based defenses to malicious models in FL. The attacker could, however, easily adapt to some obvious parameter

	Batch Size = 1	Batch Size = 8	Batch Size = 16
Length 32	`Ancient Egyptian deities Egypt the gods and goddesses worshipped. ancient gods are The beliefs of rituals surrounding these in`	`Ancient Egyptian deities are the gods and goddesses worshipped in ancient Egypt ph The beliefs and rituals surrounding these gods`	`Ancient for deities are the gods and goddesses worshipped in ancient Egypt. The beliefs and rituals surrounding these gods`
Length 128	`Ancient Egyptian deities are the gods and goddesses worshipped Egypt ancient constitu. The beliefs and rituals myths these gods`	`Ancient Egyptian deities are the gods and goddesses worshipped in ancient Egypt. The beliefs view rituals surrounding these gods`	`Ancient Egyptian deities are the gods and goddesses worshipped in ancient Egypt. The beliefs view rituals surrounding these continue`
Length 512	`Ancient Egyptian well are the gods and goddesses worshipped in ancient Egypt � The beliefs whereas ritualsies these gods formed`	`Ancient Egyptian deities are the gods and goddesses worshipped in ancient vague. " beliefs and. tried these gods`	`Ancient Egyptian deities are the gods and goddess hours thoughts in ancient final conception divine beliefs and rituals and these`

Figure 8: The first 20 tokens reconstructed from a GPT-2 model update for different combinations of sequence length and batch size. Highlighted text represents *exact* matches for both position and token to the original text (a randomly selected user).

inspection defenses. For example, simply adding a small amount of noise to the measurement vector, and other conspicuous parts of the attack, does not affect the success of the attack significantly, but does immediately make a strategy like rank inspection null and void. We confirm this adaptation by adding a small amount of Gaussian noise to each row of the measurement vector in the linear layers (for the Transformer-3 model, 8 users, 32 sequence length) and find that numerically, these layers do indeed become full rank, and the success of the attack remains largely unchanged (accuracy 90.62%). Because such simple adaptations are easily made, we do not recommend this line of defense.

Instead, we advocate for strong differential privacy on the user side. Such measures could include gradient clipping/noising. However, the imprint strategy, as found in Fowl et al. [2021], has been shown to be robust to certain amounts of gradient noise.

H Further Results

Here we present further results for our attack.

We first present qualitative results for our method. Figure 8 shows partial reconstructions for a randomly selected, challenging sequence as batch size and sequence length increase. We find that even for more difficult scenarios with more tokens, a vast majority of tokens are recoverd, and a majority are recovered in their *exact* correct position.

Comparison to other threat models. Some previous works have approached the topic of reconstructing user data from FL language model updates. These works focus on the “honest-but-curious" threat model where benign parameters are sent to the user, and the FL protocol is performed normally. We compare by fixing an architecture - in this case the Transformer-3 model described earlier. We then consider the setting with batch size 1 and evaluate our proposed method against the the TAG attack proposed of Deng et al. [2021] which improves upon previous results in Zhu et al. [2019]. We experiment for varying sequence lengths in Figure 9. The performance of TAG very quickly degrades for sequences of even moderate length. For a single sequence, our proposed attack maintains high levels of total accuracy for sequences of length exceeding 1000 tokens, whereas the total accuracy for TAG drops below 20% almost immediately, and soon after approaches 0%. Overall, we thus find that the severity of the threat posed by our attack is orders of magnitude greater than the current

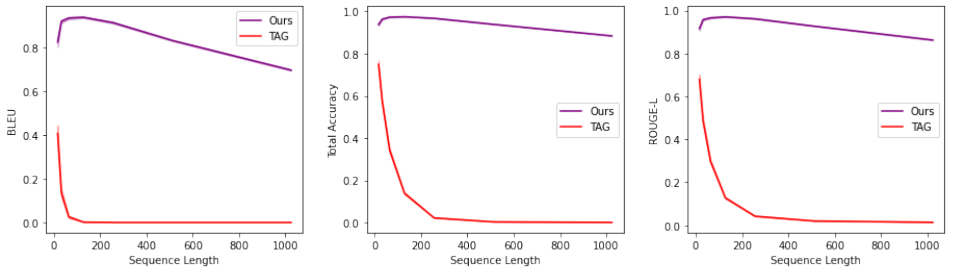


Figure 9: Comparison between our method and TAG (honest-but-curious SOTA) for the 3-layer transformer architecture described in Wang et al. [2021], variable sequence length (batch size 1).

state-of-the-art in the “honest-but-curious" threat model, and argue for the re-evaluation of the amount of privacy leaked by FL applications using transformers.

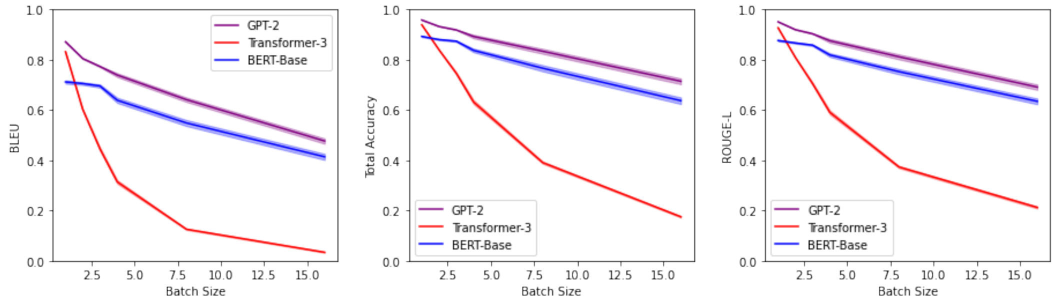


Figure 10: Baseline results for our method for different popular architectures and metrics for variable batch size (sequence length 512).

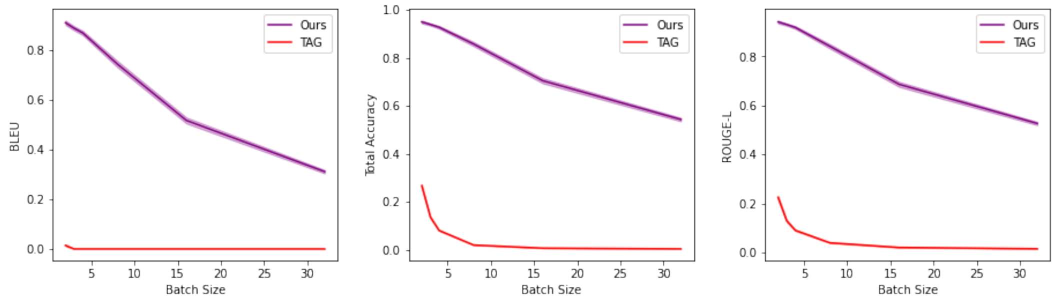


Figure 11: Comparison to TAG attack for variable batch size (sequence length 32).

I Further Figures and Explanation

In Figure 13, we illustrate the embedding recovery process that we adapt from Fowl et al. [2021]. We opt to initialize the rows of the first linear layer in each Transformer block to a randomly sampled Gaussian vector (all rows being the same vector). Then, following the calculation of Fowl et al. [2021], when an incoming embedding vector, for example, corresponding to the word "pepper", is propagated through this layer, the ascending biases threshold values of the inner product between the Gaussian vector, m, and the embedding vector, x. Because of the ReLU activation, this means that individual embedding vectors can be directly encoded into gradient entries for rows of this linear layer.

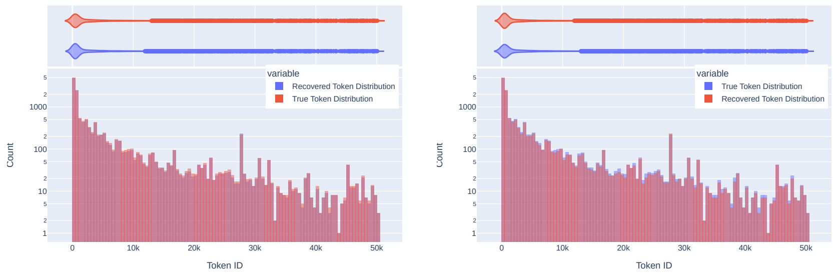


Figure 12: Bag-of-words Accuracy for from embedding gradients out of 13824 tokens from *wikitext*. **Left:** The small transformer model where tokens frequency can be estimated directly from the decoder bias. Token frequency estimation is 96.7% accurate, unique token recovery are 99.8% accurate. **Right:** The (small) GPT-2 variant where tokens are estimated from the norms of the embedding rows. Token frequency estimation is 93.1% accurate, unique tokens are 97.1% accurate due to the cutoff at $\frac{3\sigma}{2}$.

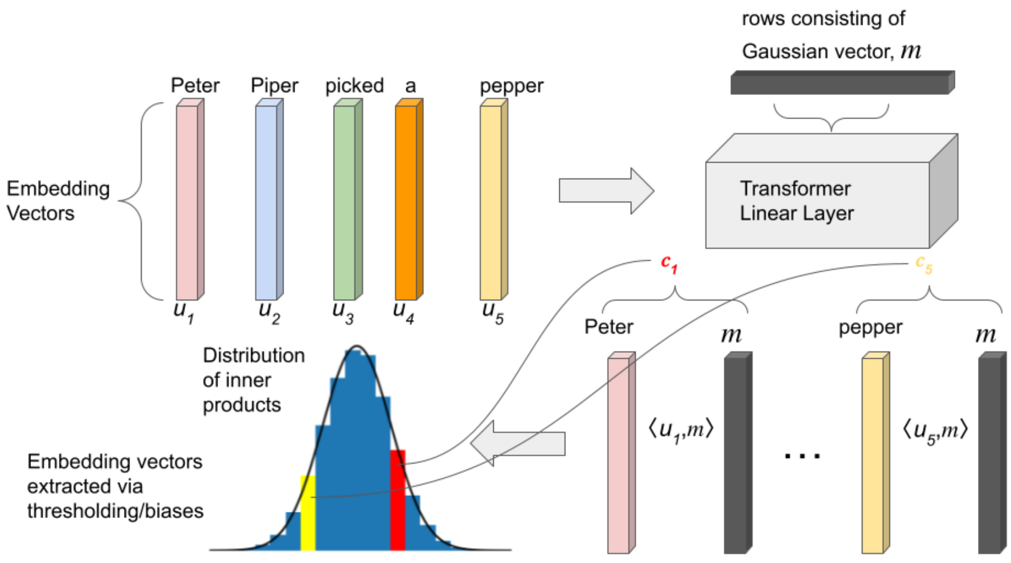


Figure 13: A visual pipeline of how embeddings are extracted from linear layers in the transformer blocks. Putting this together with the position, sentence matching strategies allows an attacker to recover user sequences.

J Additional results for the discussion period.

For ease of access, we include new results added during the discussion period here. In the camera-ready version of this document, these will be moved to appropriate sections of the appendix and main body.

J.1 Measuring Accuracy of the Best-Recovered Sentence

In the main body we present results evaluating average recovery, where we measure ROUGE, BLEU and accuracy scores on a per-sentence level and record average performance. These metrics are hence a measure of the average leakage per sentence. To get an estimate of worst-case leakage, and compare to related metrics in other attacks, we also include a measure of maximal accuracy. Instead of returning the average accuracy per sentence, we return the accuracy on the best-recovered / most-leaked / most-vulnerable sentence. This metric is visualized in Figure 14. We find that the

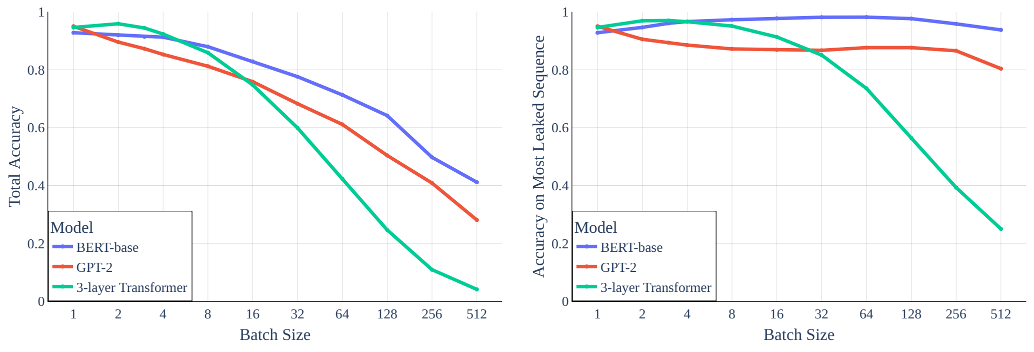


Figure 14: **Average vs worst-case Accuracy**. **Left:** Total accuracy (i.e. percentage of tokens recovered correctly in their correct position) for all considered and varying batch sizes with sequence length **32**. **Right:** Accuracy on the most-leaked sentence. Even for larger

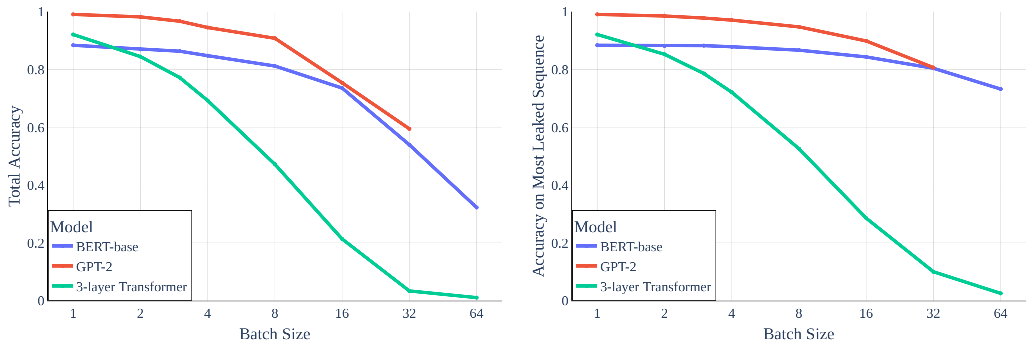


Figure 15: **Average vs worst-case Accuracy**. **Left:** Total accuracy (i.e. percentage of tokens recovered correctly in their correct position) for all considered and varying batch sizes with sequence length **512**. **Right:** Accuracy on the most-leaked sentence. Even for larger

most-vulnerable sequence continues to be near-perfectly recovered for all of the investigated range for both larger models. The much smaller 3-layer transformer, maximal accuracy drops to 30% at a batch size of 512.

J.2 Sensitivity to threshold in token estimation

We include an evaluation of the threshold parameter for the embedding-norm token estimation in Figure 16. This value is set to 1.5 standard deviations in all other experiments.

J.3 Simulated data versus public data for measurement vector estimation

As described in the main body, in all experiments in the main body we estimate the scalar mean and standard deviation of the measurement vector using public data. However, this is not necessary. These quantities can also be estimated simply using a small number of batches filled with random token ids. We show in Figure 17 that this leads to essentially the same performance in average total accuracy and to only minor differences in peak accuracy.

J.4 Other Sources of Data

In the main body we present based on *wikitext* data, split into users along wikipedia articles. In Figure 18 we show the same attack for other sources of text data, namely the `shakespeare` dataset described in Caldas et al. [2019] and the `stackoverflow` dataset as used in Wang et al. [2021]. We also include a data source with a stream of entirely random tokens. Ultimately, the actual text distribution matters little for the attack proposed here, all investigated sources of real text are equally vulnerable. Random text is noticeably more vulnerable. This could be problematic from a security

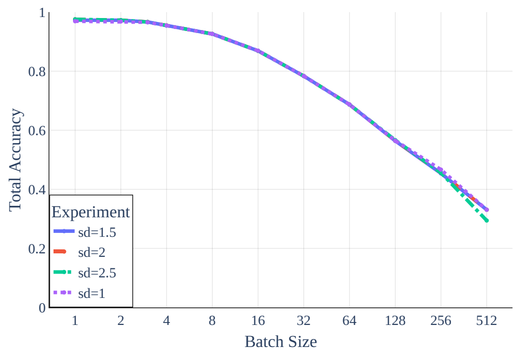


Figure 16: **Sensitivity to token estimation threshold.** Total accuracy (i.e. percentage of tokens recovered correctly in their correct position) for GPT-2 and varying batch sizes with sequence length 32. We evaluate the threshold parameter for the token estimation. The default value in the remainder of all experiments is 1.5, although we find the estimation to be robust to a range of similar cutoffs.

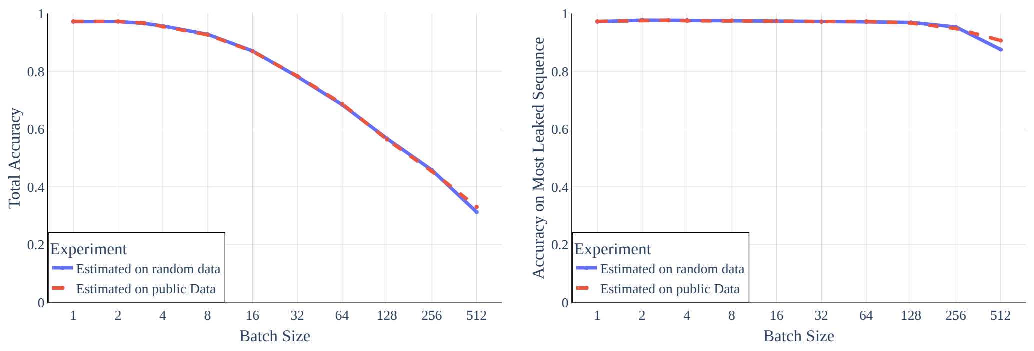
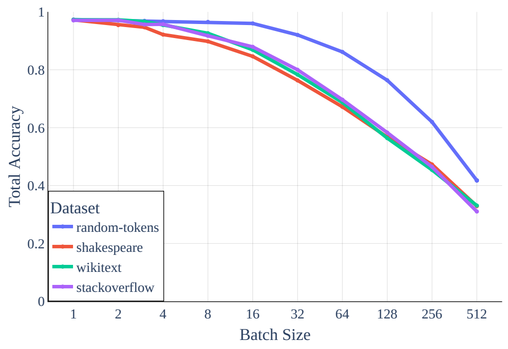


Figure 17: **Using random data to estimate the measurement distribution**. **Left:** Total accuracy (i.e. percentage of tokens recovered correctly in their correct position) for GPT-2 and varying batch sizes with sequence length 32. **Right:** Average token accuracy on the most-leaked sentence. Random tokens can be an effective substitute for public data to estimate the measurement mean and standard deviation.

perspective, as an attacker would often be interested in text fragments that are not usual text, e.g. passwords or social security numbers.

J.5 Total Amount of Token Leaked

Finally, in Figure 19 we include the total amount of tokens leaked correctly through the attack for a variety of sequence lengths, batch sizes and models. We see that the total amount of tokens leaked with GPT-2 has not yet reached a peak, and larger sequences could leak more tokens. For the smaller 3-layer transformer we see that the number of leaked tokens peaks around a batch size of 128 for a sequence length of 32, and 8 for a sequence length of 512, i.e. around 4096 tokens. This is related to the number of bins in this architecture, which in turn is given by the width of all linear layers, leading to a bin size of 4608 for the 3-layer transformer. For the small GPT-2 the number of bins is 36864.



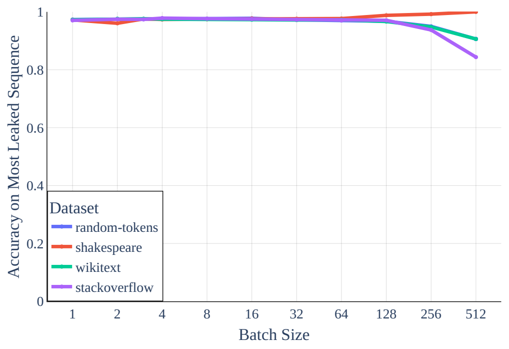
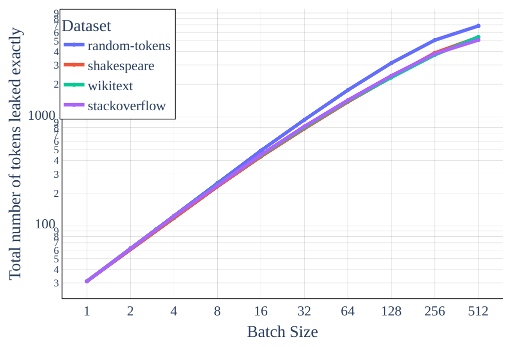
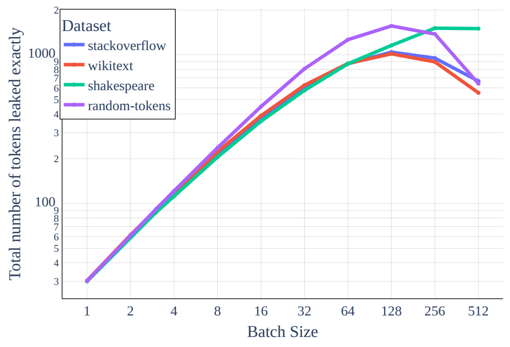
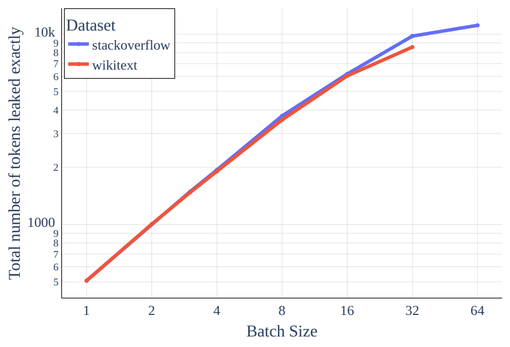


Figure 18: **Attacking other data sources**. **Left:** Total accuracy (i.e. percentage of tokens recovered correctly in their correct position) for GPT-2 and varying batch sizes with sequence length. **Right:** Average token accuracy on the most-leaked sentence. The actual data source matters little for the attack and all sources of real data are almost equally vulnerable. Random data is more vulnerable in average case, due to fewer collisions. Shakespeare data is most vulnerable in the maximum case, due to the existence of some very simple sentences for most speakers/users.







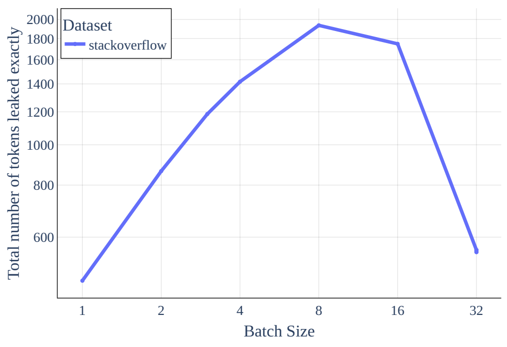


Figure 19: **Total amount of tokens leaked in their exact location for various settings.** Top row: Total number of token for GPT-2 (left) and the 3-layer transformer (right) for a sequence length of 32 and various batch sizes. Bottom row: otal number of token for GPT-2 (left) and the 3-layer transformer (right) for a sequence length of 512 and various batch sizes.