PANNING FOR GOLD IN FEDERATED LEARNING: TARGETED TEXT EXTRACTION UNDER ARBITRARILY LARGE-SCALE AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

As federated learning (FL) matures, privacy attacks against FL systems in turn become more numerous and complex. Attacks on language models have progressed from recovering single sentences in simple classification tasks to recovering larger parts of user data. Current attacks against federated language models are sequence-agnostic and aim to extract as much data as possible from an FL update - often at the expense of fidelity for any particular sequence. Because of this, current attacks fail to extract any meaningful data under large-scale aggregation. In realistic settings, an attacker cares most about a small portion of user data that contains sensitive personal information, for example sequences containing the phrase "my credit card number is ...". In this work, we propose the first attack on FL that achieves targeted extraction of sequences that contain privacy-critical phrases, whereby we employ maliciously modified parameters to allow the transformer itself to *filter* relevant sequences from aggregated user data and encode them in the gradient update. Our attack can effectively extract sequences of interest even against extremely large-scale aggregation.

1 INTRODUCTION

Machine learning models continuously grow in size and require ever more data to improve. This paradigm of scaling has lead to significant improvements in a multitude of applications, but it also exerts an ever increasing toll on privacy. To train large models, data has to be collected and centralized by single parties, for example, tech companies with large enough user bases. As such, these entities are interested in collecting as much data as possible and in this way extract value from their users. Moreover, once data is collected, it is rarely erased. This is fundamentally in conflict with commonly held notions of privacy.

Against this backdrop, federated learning has emerged as the field of core technologies aiming to train machine learning models with *decentralized data*, that is without the need for a central party to collect all data. By exchanging model updates instead of data, user devices update a centralized model without ever giving up control of their data. This concept has made federated learning (FL) a popular direction for the future of distributed computing with privacy. In many applications, federated learning is slower than centralized training (Bonawitz et al., 2019), but benefits to privacy may outweigh these limitations, especially in applications like next-word keyboard prediction that handle all text that a user might type in a smartphone (Hard et al., 2019).

In turn, this expectation of privacy through federated learning is sometimes taken for granted. In reality, the actual privacy achieved by federated learning systems depends on a large number of factors and parameters - model sizes, architectures, user numbers, aggregation amounts, among others. Attacks against privacy in federated learning probe this boundary, empirically discovering which federated learning scenarios are unsafe (Phong et al., 2017; Melis et al., 2019; Geiping et al., 2020).

In this work, we are particularly interested in federated learning systems involving transformer architectures (Vaswani et al., 2017) which form the backbone of many recent breakthroughs with large models (Brown et al., 2020; Dosovitskiy et al., 2021; Jumper et al., 2021), and especially applications in text, which represent a key point of interest in many modern applications of federated learning (Paulik et al., 2021; Dimitriadis et al., 2022). Our main threat models of interest here are

untrusted server scenarios, also known as *malicious server* scenarios, contrasting themselves with investigations in *honest-but-curious* scenarios, which limit the extent of possible attacks. *Untrusted server* scenarios are of critical importance from a user-centric privacy perspective. If a federated learning system is supposed to uphold privacy, then ideally this privacy can be guaranteed without having to assert full trust in the server. After all, a perfectly trustworthy server could run the simplest FL protocol: Download all user data, promise not to copy it, train a centralized model, and delete all data. Another way to look at *untrusted server* scenarios is to see them as a worst-case scenario. Even if we believe a server will uphold privacy, we might wonder about the worst-case loss in privacy that would arise if this server is even briefly compromised (through either classical security breaches or poisoning attacks (Bagdasaryan et al., 2019)) and acts maliciously.

A central idea for such a malicious server is to modify the current state of the partially trained machine learning model that is shared with the users. What is the worst-case model state that a malicious server can send to users? As this model directly operates on user data according to the chosen FL protocol, this can be understood as an analogue to *untrusted code instructions* that are being evaluated on a user's private data (OWASP, 2022; Fowl et al., 2022).

In this work, we discuss a novel attack threat where a malicious server is able to extract key fragments of private information even from industrial-sized streams of data. This novel attack, which we dub "panning" as it sifts through large amounts of text, is capable of targeting key phrases, such as `credit card` or `social security number` and extracting all tokens of user data that follow the occurrence of this trigger. This is an essential shift in capabilities for attacks against transformer-based model in federated learning. Existing attacks only recover user data - no matter how private or benign - in scenarios where the number of model parameters is significantly smaller than the number of tokens in a user update (Fowl et al., 2022; Gupta et al., 2022; Dimitrov et al., 2022; Pasquini et al., 2021). In some cases, attacks can siphon random examples of user data but only through a large number of repeated queries (Wen et al., 2022). In contrast, the discussed attack has the capability to target key phrases of private user data and recover sensitive text, e.g. credit card or social security numbers, or any other secrets, following these triggers. In comparison to existing attacks, this attack is also effectively independent of the amount of user data contained in an update and does not degrade as many user updates are securely aggregated (Bonawitz et al., 2017).

2 Background and Application Examples

Applications in text have been among the first systems where federated learning has seen use in industrial settings such as keystroke prediction in Hard et al. (2019); Ramaswamy et al. (2019), settings search in Bonawitz et al. (2019), news personalization (Paulik et al., 2021), and improved messenger services on Android (Google, 2022). To discuss additional details for the last example, the documentation of Google messages[1], describes that federated learning is used to improve ML models related to Google messages, for example "smart reply" features. FL is active by default on all Android devices with this app. According to the documentation, user conversations are protected only through secure aggregation (Bonawitz et al., 2017), which "*can't reveal your conversations or content to Google or anyone else [...], grouping many similar adjustments together so that Google can't inspect an adjustment from a single device*". In this work, we argue that this system of protecting privacy leaves users open to targeted extraction attacks from a malicious server update sent to the user, if the system uses a transformer-based machine learning model.

Previous attacks against privacy in federated learning with language models, in comparison, are not capable of breaking this application. As discussed, we group attacks according to their threat model, as either "honest-but-curious", or "malicious". The earliest attacks against transformer models in Zhu et al. (2019b) and improvements in Deng et al. (2021) operate in the "honest-but-curious" threat model. These attacks have resulted in only limited success, recovering data only from single sequences. In practice, "single sequences" imply that user updates are not securely aggregated, a single user exists and this single user sends back an update based on only one sequence of text. Recent developments in Dimitrov et al. (2022) and Gupta et al. (2022) have pushed the envelope, reconstructing more and more text fragments, even from multiple sequences in this restricted threat model. On the other side, malicious server threat models, for example in Pasquini et al. (2021)

[1] `https://support.google.com/messages/answer/932790`

and Fowl et al. (2022) can recover more information and identify or respectively reconstruct from aggregates of hundreds of sequences.

These attacks all attempt to recover every piece of user data used to compute the aggregated model update, putting equal emphasis on the fidelity of each reconstructed sequence. This strategy inevitably decreases the capability of previous attacks to recover accurate individual sequences as the number of sequences and tokens increases, and eventually the extracted sequences become meaningless combinations of tokens.

Attacks attempt to break aggregation outside this format generally require additional assumptions. For example, Lam et al. (2021) shows that aggregation can be breached if side-channel metadata is available to the server. Pasquini et al. (2021) show that if users can be addressed separately by the server and each user only has limited data (e.g. a single sequence), malicious parameters with zero-gradients can be used to suppress all other users and break aggregation. This trick can also be used for targeted membership inference, checking if a user owns a specific datapoint. Further, Wen et al. (2022) shows that if malicious servers are allowed to send multiple queries, then they can "fish" for single data points, even from aggregated data.

Yet, in a real-life scenario, among all data a user computes and aggregates model updates with, often only a few of them contain information valuable to a potential attacker. For example, out of a 50-sentence online conversation between a user and a bank agent, an attacker may only be interested in the particular sentence that contains the user's social security number. This poses the question of whether it is possible for an attacker to dedicate the entire capacity of their breaching algorithm to only a limited number of target sequences with a specific set of keywords or triggers.

Threat Model - Untrusted Server Our threat model contains two parties. First, a user (or group of aggregated users) that owns text, which contains private information following a set of keywords $K = \{k_1, \ldots, k_n\}$. Second, a malicious server that aims to "pan", to perform targeted extraction of all tokens of user data following that same set of keywords $K = \{k_1, \ldots, k_n\}$. We assume that the number of aggregated sequences may be unlimited, but the number of sequences matching the combination of keywords is limited. We assume that the federated learning exchange is otherwise secure: Both parties agree beforehand on a transparent implementation for both model architecture and user-side protocol that is vetted by public examination. The only attack vector for the server is the model update sent indiscriminately to all users in the group.

Different stronger threat models have also been investigated in literature (Frey, 2021; Boenisch et al., 2021; Fowl et al., 2021). Threat model considered in Frey (2021) allows server to execute arbitrary pieces of code on user device, while Fowl et al. (2021); Boenisch et al. (2021) assumes a server can maliciously modify model architectures. We also note that the role of the attacker needs not be played by the party that owns the server, and any individual/party that has access to the model update anywhere in the pipeline can assume the role. Particularly, an attacker can inject the attack into incoming model parameters for users, retrieve the resulting unsecured user update, and replace the outgoing update with seemingly normal model gradients.

Remark (Peer-to-Peer Attacks between Users.)**.** *Further, there are even cases where this threat model applies even if the server is fully benign: If in any round of training, a group of malicious users can control enough updates to replace the model parameters (Bagdasaryan et al., 2019; Bhagoji et al., 2019) in a single round of training, then the server will unwittingly distribute these malicious model parameters in the next round of training, collect model updates from other, vulnerable users, aggregate them and redistribute the updated model. Because an attack through "panning" is invariant to aggregation sizes, this means that the group of malicious users can then discover data from vulnerable users in the next round of training, while the server remains unaware they have been used to breach privacy.*

3 METHOD

Here we discuss "panning" attack that performs targeted extraction of "relevant" sequences that contain a specific set of keywords. These attacks are motivated by previous unstructured attacks on transformer architectures (Fowl et al., 2022), and the key ideas are:

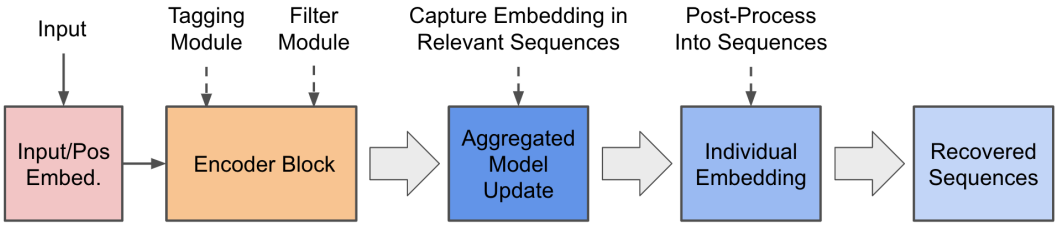

Figure 1: An overview of our proposed panning attack. In particular, an attacker reprograms the encoder block with a tagging module and a filter module, which encodes keyword-specific information into model gradient. The information enables the attacker to pan for individual relevant tokens from aggregated user update. The attacker can then post-process the retrieved tokens to recover original sequences.

1. Modification of multi-head attention blocks (MHA) to "tag" the tokens of a sequence that contains specific keywords, as in Section 3.2.
2. Modification of linear layers to "filter out" tokens in irrelevant (i.e. non-target) sequences, as in Section 3.3.
3. Leveraging the tag produced by the tagging module to capture tokens embedding belonging to target sequences, as in Section 3.3.
4. Post-processing the relevant embeddings into target sequences, as in Section 3.4.

An overview of our attack algorithm is presented in Figure 1.

3.1 Overview

As discussed in Phong et al. (2017); Geiping et al. (2020); Fowl et al. (2021), given a linear layer parameterized by $(\mathbf{W}, \mathbf{b})$ whose forward pass is $y = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$, an attacker can recover inputs to this layer from the aggregated gradient as

$$\sum_j \mathbf{x}_j = \sum_j \nabla_{W^m} L_j \oslash \sum_j \frac{\partial b^m}{\partial L_j}, \tag{1}$$

where $L_j = L(x_j; \mathbf{W}, \mathbf{b})$, $\oslash$ stands for element-wise division, and W^m, b^m stands for any m-th row and m-th entry of $\mathbf{W}$ and $\mathbf{b}$ respectively. Based on this observation, Fowl et al. (2021) proposed malicious modifications of $\mathbf{W}, \mathbf{b}$, which together with ReLU activation, allows an attacker to recover individual $\mathbf{x}_j$ - even from update averaged over many users. This technique was employed in the domain of image classification.

However, as laid out in the following equation, adapting this trick to transformer presents additional challenges. In particular, transformer mixes token embedding and positional embedding, and adds the results of multi-head self-attention (MHA) on top of the mixed embedding before forwarding through the earliest linear layer. In fact, given sequence length ℓ and number of sequences B, examining the aggregated gradient of the earliest linear layer reveals that

$$\sum_{i=1}^{B}\sum_{j=1}^{\ell} \nabla_{W_m} L_j \oslash \sum_{i=1}^{B}\sum_{j=1}^{B} \frac{\partial b_m}{\partial L_j} = \sum_{i=1}^{B}\sum_{j=1}^{\ell} f_{i,j} = \sum_{i=1}^{B}\sum_{j=1}^{\ell} e_{i,j} + p_j + \texttt{MHA}(\{e_{i,j'} + p_{j'}\}_{j'=1}^{\ell}; j)$$

Here $e_{i,j}$ is the embedding of the j-th token in the i-th sequence, p_j is the j-th positional embedding, and $\texttt{MHA}(\{e_{i,j'} + p_{j'}\}_{j'=1}^{\ell}, j)$ is the j-th entry after applying MHA on sequence i. To recover meaningful sequence from the aggregated mixed embedding $\sum_i \sum_j f_{i,j}$, an attacker needs to (1) isolate out individual mixed embedding $f_{i,j}$, (2) relate $f_{i,j}$ to the sequence i and position j, and (3) match $e_{i,j}$ to the most probable word.

Previous work (Fowl et al., 2022) proposed an attacking algorithm that attempts to overcome these challenges. However, their attack focuses on recovering as much data as possible, and therefore the fidelity of individual recovered sequences degrades as the number of sequences increases. On the

other hand, attackers often only care about the few sequences that contain privacy-critical information, and are happy to forego fidelity of irrelevant sequences in exchange for accurate recovery of valuable sequences. Based on this philosophy, we propose a novel attacking algorithm that filters out irrelevant mixed embedding, enabling an attacker to pan for privacy-relevant sequences even out of extremely large-scale aggregation.

3.2 TAGGING TARGET SEQUENCES

In this section, we describe how our attack employs malicious modifications of transformers to "tag" the mixed embedding that belongs to a sequence that contains the selected keywords $K = \{k_1, \ldots, k_N\}$. We illustrate the modification in Figure 2. For simplicity, we start by detailing this construction with only one keyword k.

To facilitate the construction, we assume the norm of token embedding $e^{(k)}$ associated with the target word k to be larger than the norm of all other token and positional embeddings. The server can easily meet this assumption as both embedding matrices are also part of the model parameters under their control. By inspecting the transformer architecture, it is immediately clear that tokens in the same sequence interact with each other only in the MHA block. Therefore, we design a self-attention mechanism that tags the target sequences by imprinting $e^{(k)}$ into every mixed embedding in relevant sequences. To be more specific, let $(\mathbf{W}_Q, \mathbf{b}_Q), (\mathbf{W}_K, \mathbf{b}_K), (\mathbf{W}_V, \mathbf{b}_V)$ be the weight matrices and biases of query, key, and value layers respectively. We set each of the parameters as follows,

$$\begin{cases} \mathbf{W}_Q = \mathbf{0}, & \mathbf{b}_Q = \alpha e^{(k)}, \\ \mathbf{W}_K = \mathbf{I}_d, & \mathbf{b}_K = \mathbf{0}, \\ \mathbf{W}_V = \mathbf{T}_{d'}, & \mathbf{b}_V = \mathbf{0}, \end{cases} \quad \mathbf{T}_{d'}[q, r] = \begin{cases} 1 & q + d' = r, q \leq d', \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where α is a large positive value (e.g. 10^8) and $\mathbf{T}_{d'}$ is the truncated shift matrix with pre-defined dimension d'. The construction of $\mathbf{W}_K$ and $\mathbf{b}_K$ leaves the key $\mathbf{K}$ unaltered. The construction of $\mathbf{W}_Q$ and $\mathbf{b}_Q$ produces the query Q with exactly the same rows $e^{(k)}$. Finally, the construction of $\mathbf{W}_V$ and $\mathbf{b}_V$ "shifts" the entries block indexed by $[d' : 2d']$ of each input embedding to the first d' positions, and sets all other positions to zeros.

Next, we evaluate the results of the MHA computation based on this construction. If a length-ℓ sequence $\{e_j\}_{j=1}^{\ell}$ contains the target word k at position j', then the attention weight calculated by $\texttt{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})$ will favor attending every single word in the sequence to the target word. In fact, when α is large enough, the softmax transforms the attention weight into a Dirac-delta function, making attention to the target word absolute. Consequently, the resulting mixed embedding after the MHA block becomes:

$$f_j = e_j + p_j + \mathbf{T}_{d'}(e^{(k)} + p_{j'}) \tag{3}$$

Or, in plain language, the entries d' to $2d'$ of the target embedding are imprinted to the first d' position of every embedding in the sequence. Interestingly, in the case of causal language model, applying construction in Equation (2) on masked MHA turns the keyword into a trigger, where the module only imprints $e^{(k)}$ to tokens that appear *after* the keyword in the relevant sequences. Coupling with filter module introduced later in Section 3.3, one can instead perform targeted extraction of *sub-sequences* that start with k. On the other hand, if the sequence does not contain the keyword k, or in the case of masked MHA, if the mixed embedding appears before k, then the imprinted embedding will be randomly picked from e_1 to e_ℓ. The difference allows our malicious modification to tag the mixed embedding in target sequences by a unique imprint, and we will discuss in the next section how we filter out irrelevant sequences (not containing k) based on this imprint.

We note that extending the above construction to multiple keywords is straightforward. In particular, for N keywords $k_1, \ldots, k_N$, one can simply repeat the above constructions on different head in MHA block for different keywords, and imprint each keyword-specific signature to different positions of the mixed embedding. We also note that Fowl et al. (2022) employed a related construction technique, with the goal of imprinting a part of e_1 onto every token $e_1, \ldots, e_\ell$ in the same sequence. This position imprinting enables grouping tokens into sentences and can easily be used as conjunction with our tagging module.

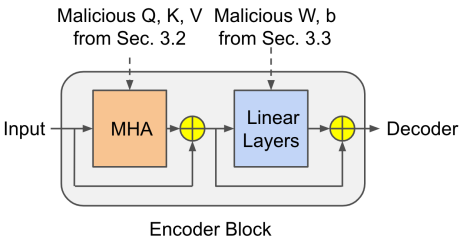

Figure 2: An illustration of our key modifications. We maliciously modify the query, key and value layer in the MHA block to tag the mixed embedding of relevant sequences. We also modify the feed-forward layers to encode statistically distinguishable signatures to each token based on the tag. The signature is then used to filter out irrelevant mixed embedding. Note that the $\oplus$ represents add-and-norm operation in transformer.

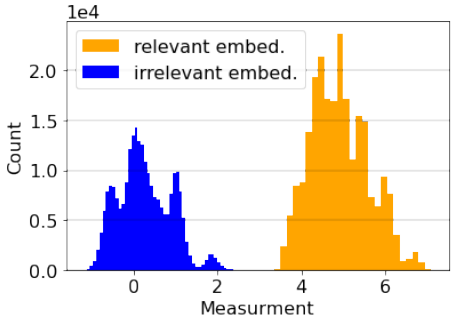

Figure 3: The measurement distributions of mixed embedding by $\mathbf{w} + \mathbf{w}'$ from Section 3.3 on GPT-2. The measurement distribution of irrelevant mixed embedding (blue) exhibits clear separation from the measurement distribution of relevant mixed embedding (orange). We set $\beta = 2.5$ here.

3.3 Filter out irrelevant tokens

Next, we discuss our approach to filter out irrelevant sequences based on the tag of each token, and provide an illustration of the key idea in Figure 2. Our approach draws inspiration from Fowl et al. (2021), which attempted to perform gradient separation on input uncovered in Eq. 1. The authors construct a malicious linear layer by assigning to each row of $\mathbf{W}$ identical copies of a measurement vector $\mathbf{w}$, and setting each entry of $\mathbf{b}$ such that every $b^m < b^{m+1}$. The authors then observe that if the linear layer is followed by a ReLU activation, one has

$$(\sum_j \nabla_{W^m} L_j - \sum_j \nabla_{W^{m-1}} L_j) \oslash (\sum_j \frac{\partial b^m}{\partial L_j} - \sum_j \frac{\partial b^{m-1}}{\partial L_j}) = \sum_j g_m(\mathbf{w}^\top \mathbf{x}_j)\mathbf{x}_j,$$

where $g_m(z) = 1$ if $b_{m-1} \leq z < b_m$ and $g_m(z) = 0$ otherwise. The observation implies that $\mathbf{x}_j$ can be recovered exactly if only $\mathbf{x}_j$ satisfies $g_m(\mathbf{w}^\top \mathbf{x}_j) = 1$, Therefore, if the attacker can accurately estimate the distribution of $\mathbf{w}^\top \mathbf{x}_j$, they can then pick $b_1, \ldots, b_M$ accordingly to maximize the chance for every $\mathbf{x}_j$ to fall uniquely in one of the bin (b_{m-1}, b_m).

To apply the idea from Fowl et al. (2021) to transformer-based attack, Fowl et al. (2022) discovered that modeling the distribution of $\mathbf{w}^\top(e_{i,j} + p_j)$ as Gaussian works well empirically if every entry of $\mathbf{w}$ is a random sample from standard Gaussian. Fowl et al. (2022) therefore proposes to set $b_m = -\Phi^{-1}(\frac{m}{M})$ where $\Phi^{-1}(\cdot)$ is the inverse of estimated Gaussian distribution, by which each bin (b_{m-1}, b_m) approximately contains the same probably mass. However, observe that the number of bins upper-bound the capability of an attacker to recover user data. Therefore, the algorithm proposed in (Fowl et al., 2022) fails to recover meaningful sequences from large-scale aggregation as they attempt to separate an increasing number of mixed embeddings using a fixed number of bins.

To address the issue, we propose a set of malicious modifications to pan for relevant mixed embeddings. To be more concrete, we scale the norm of all embedding to 1, except for $e^{(k)}$, the embedding of the given keyword, to $\beta > 1$, and set the first d' entries of each embedding to 0. β controls the "signal strength" of relevant mixed embedding, and helps the differentiation with irrelevant embedding as detailed later. We also sample the random measurement vecotr $\mathbf{w}$ from standard Gaussian and mask out the first d' entries. Finally, we construct $\mathbf{w}'$ as

$$\mathbf{w}'[q] = \begin{cases} e^{(k)}[q + d'], & q \leq \text{d'} \\ \mathrm{N}(0, \mathbf{I}), & q > \text{d'}. \end{cases}$$

By instead setting each row of $\mathbf{W}$ to $(\mathbf{w} + \mathbf{w}')$, we have

$$(\mathbf{w} + \mathbf{w}')^\top f_j = \mathbf{w}^\top(e_j + p_j) + (\mathbf{w}')^\top \mathbf{T}_{d'}(e_{j'} + p_{j'}) \tag{4}$$

by Eq. 3 where j' is the position the MHA attends to. As a consequence, the distribution of measurement of the mixed embedding in irrelevant sequences remains zero-mean, while a mean shift of $(\beta\|\mathbf{w}'\|)^2$ can be observed in the measurement distribution of mixed embedding in relevant sequences. We also empirically verify the distinction between two distributions, and present the results in Figure 3. The distinction allows us to apply the binning strategy from Fowl et al. (2022) but instead on the relevant distribution, which effectively condenses irrelevant mixed embedding into a small number of bins. The majority of bins can then be dedicated to retrieving individual relevant mixed embedding, greatly improving the effectiveness of downstream post-processing.

3.4 Post-process mixed embeddings into meaningful sequences

With individual mixed embedding $f_{i,j}$ isolated out from user update, we then need to associate each $f_{i,j}$ with the corresponding sequence index i, position index j and the actual word.

To associate a mixed embedding $f_{i,j}$ with the corresponding sequence index i, we utilize the position imprinting module proposed by Fowl et al. (2022), which shares similar construction techniques with our tagging module as discussed in Section 3.2. To be more concrete, we construct a positional imprinting module using a different head in the MHA block as follows,

$$\begin{cases} \mathbf{W}_Q = \mathbf{O}, & \mathbf{b}_Q = \alpha p_0, \\ \mathbf{W}_K = \mathbf{I}_d, & \mathbf{b}_K = \mathbf{0}, \\ \mathbf{W}_V = \mathbf{T}_{d'}, & \mathbf{b}_V = \mathbf{0}, \end{cases} \quad \mathbf{T}_{d'}[q,r] = \begin{cases} 1 & q + d' = r, 2d' < q \leq 3d', \\ 0 & \text{otherwise.} \end{cases}$$

For a given sequence of embedding $\{e_j\}_{j=1}^{\ell}$, the resulting $\{f_j\}_{j=1}^{\ell}$ after going through the positional imprinting module is calculated as

$$f_j = e_j + p_j + \mathbf{T}_{d'}(e_1 + p_1), \tag{5}$$

where sentence-specific information is imprinted to entries $2d + 1$ to $3d$ of each mixed embedding. We then perform constrained K-means clustering (Bradley et al., 2000) over all recovered $f_{i,j}$ based on the sentence-specific entries, where the number of centers is the same as the number of total sequences, and each center can at most be associated with ℓ mixed embedding.

After grouping mixed embeddings into sequences, the next step is to recover the position. As mixed embeddings in the same sequences are guaranteed to have distinct positions, we perform position recovery on a sequence-by-sequence basis. For a set of mixed embedding $\{f_{j'}\}$ grouped into the same sequence, identifying the most suitable unique position j of each $f_{j'}$ can be viewed as a maximum weight bipartite matching problem, where we define the weight of a potential match $(f_{j'}, j)$ to be the correlation between $f_{j'}$ and p_j. The problem can equivalently be transformed into a linear assignment problem (Kuhn, 1955), which is identical to the formulation discussed in Fowl et al. (2022), and solved efficiently. We can then obtain the pure token embedding $e_{i,j}$ by subtracting the recovered positional embedding from $f_{i,j}$.

The final step is to associate each recovered token embedding $e_{i,j}$ with the actual words. Ideally, for each $e_{i,j}$ one can greedily search over every actual token embedding $\{e_1, \ldots, e_V\}$ to find the word whose associated token correlates with $e_{i,j}$ the most. On the other hand, Fowl et al. (2022) discovered that the frequency of tokens presented in user updates can be estimated accurately. The estimated frequency dictates the maximum number of $e_{i,j}$ a specific word can be associated with, which can be used as a constraint to formulate another maximum weight bipartite matching problem, where the weight of a potential match $(e_{i,j}, e_v)$ is again the correlation between two embeddings. Compared to greedy search, the additional constraint allows some imperfectly recovered token embedding to still be matched with correct words.

4 Empirical Evaluation of the Attack

In this section, we empirically evaluate our proposed attack on different transformer architectures commonly used in real-world applications. In particular, we consider the smallest variant of GPT-2 (Radford et al., 2019) with 124 million parameters, which is a causal language model with masked self-attention. We also consider a variant of BERT (Devlin et al., 2019) with 110 million parameters, which is used in previous works (Deng et al., 2021; Zhu et al., 2019a) for attack evaluation. Unlike GPT-2, BERT is trained as a masked language model which allows bidirectional self-attention.

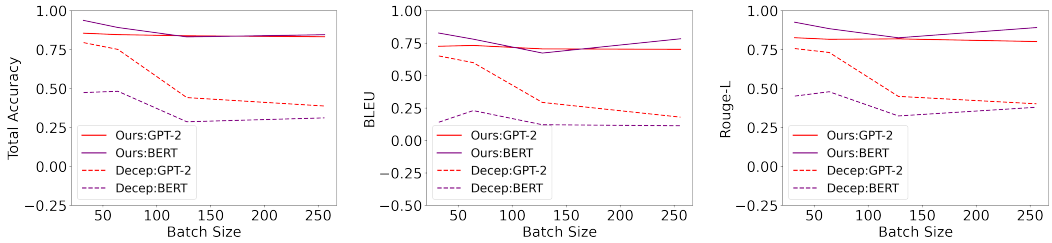

Figure 4: Comparison of our method with Decepticons, a recent Transformer-based attack, for different architectures across various batch sizes. The results of Decepticon are depicted in dased lines. The sequence length is fixed to 32. Our attack outperforms Decepticons for every batch size, and the performance remains stable as batch size increases.

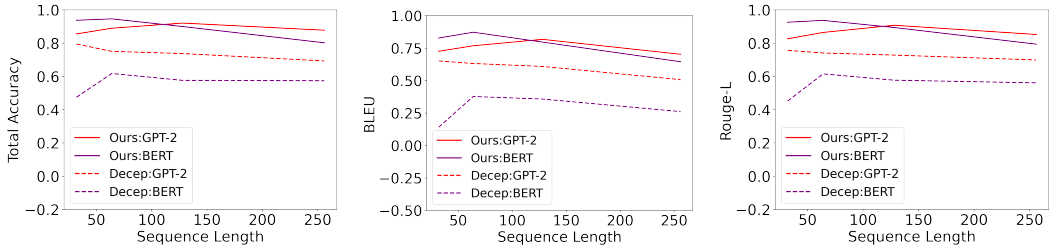

Figure 5: Comparison of our method with Decepticons for different architectures across various sequence lengths. The batch size is fixed to 32. We evaluate both methods only on the target sequences. Our attack remains agnostic to sequence length, and outperforms Decepticons in all metrics on target sequences.

We evaluate our method on the *wikitext* dataset (Merity et al., 2016). We partition the dataset into articles and assign each user one article as their private data. We focus on fedSGD, where each user performs a single gradient step and sends the model update to sever. We perform quantitative evaluation on our method with a range of different metrics. Specifically, we consider the widely adopted BLEU score (Papineni et al., 2002), the ROUGE-L (Lin, 2004), and total accuracy (Fowl et al., 2022). We provide additional details in Appendix A.

4.1 Comparing with Transformer-based Attacks for Malicious Servers

We begin by comparing our proposed method with Decepticons (Fowl et al., 2022), a recent transformer-based attack for the malicious server threat model, across various batch size and sequence lengths. To simulate the scenario where only a small number of sequences contain privacy-critical information, we select the first 3 sequences from each batch and replace one of the tokens of each sequence with the token of a target keyword. For experiments on BERT, the replaced token is randomly selected. For experiments on GPT-2, we instead replace one of the first four tokens with trigger keyword, which allows us to assess the impact of imperfect imprints created by masked self-attention. We note as GPT-2 performs causal language inference, an attacker targeting related tasks generally cares less about tokens that appear before the trigger, and hence placing the keyword further back in the sequence is unnecessary. The quantitative results are then evaluated only on these target sequences.

We first fix the sequence length, and compare our method with Decepticon using varying batch size in Figure 4. We see that the performance of Decepticons drops significantly as the batch size increases over 128. On the other hand, our attack maintains steady reconstruction quality for recovered relevant sequences even when the batch size increases.

Then, we fix the batch size and compare across sequence length in Figure 5. The figure shows that our proposed attack also outperforms Decepticons across all combinations of model and sequence length (although the performance of Decepticons is less sensitive to the change of sequence length).

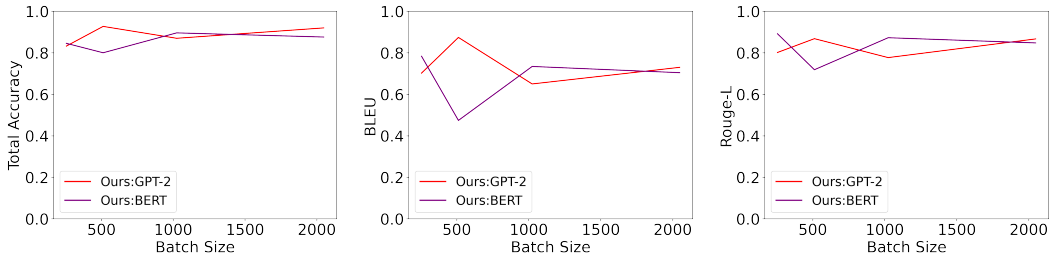

Figure 6: Experiment on our attack across different architectures for large batch size. The sequence length is fixed to 32. We evaluate our method only on the target sequences. Our attack steadily obtains high fidelity recovery even against extreme-scale aggregation. Attack success remains basically constant as batch sizes increases.

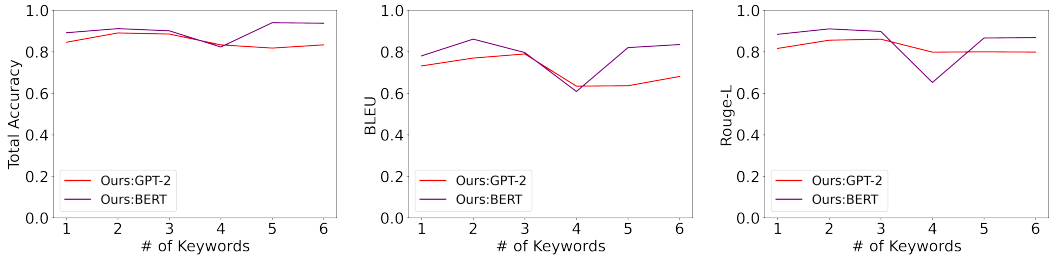

Figure 7: Experiment on our attack across different architectures for target phrases with different length. We fix the sequence length to 32 and batch size to 64 respectively. Our method is able to accurately capture relevant sequences given target phrases of various lengths. Attack success remains constant as batch sizes increase.

4.2 Experiments with Large-Scale Aggregation and Multiple Keywords

Next, we evaluate our proposed methods under more extreme conditions. First, we assess the capability of our method to capture relevant sequences under extremely large-scale aggregation. We fix the sequence length to 32, and experiments with batch size up to 2048, and summarize the results in Figure 6. As demonstrated in the figure, the fidelity of our recovered relevant sequences remains high and is largely agnostic to batch size. The observation validates the design of our proposed tagging and filtering module, which enables our attack to accurately capture tokens in relevant sequences even against extremely large-scale aggregation. We note that the results of Decepticons are not included as it becomes computationally expensive to handle this number of sequences under their method.

Additionally, we investigate how the number of keywords K impacts the performance of our attack. We perform experiments on $K \in \{1, 2, 3, 4, 5, 6\}$, For each K we create K-word-long target phrases, and replace K consecutive tokens in each of the first 3 sentences. The result is presented in Figure 7, and demonstrates that our attack is able to recover sequences relevant to a large number of keywords. We note that the number of keywords our attack can target is only limited by the number of heads in the MHA blocks, and both GPT-2 and BERT come with 12 heads, which is more than enough to cover most privacy-sensitive phrases in real world.

5 Conclusion

In this paper, we describe a vulnerability of transformers used in a federated setting. This vulnerability opens the door to an attack that can "pan" for sequences that contain private information based on specified keywords. The attack allows an adversary to accurately capture sentences that contain the keywords out of aggregated updates from thousands of sequences. The attack injects malicious parameters into the transformer whereby sensitive sequences are "tagged" with a unique signature, and filtered to recover these targeted sequences. This attack reflects a notable shift in the capabilities of data reconstruction attacks in federated learning as the attack succeeds under seemingly arbitrary amounts of aggregation.

ETHICS AND MITIGATIONS STATEMENT

We describe an attack that significantly extends the capabilities of data extraction attacks for transformer models in federated learning. However, this attack only extends the capabilities of a malicious server and furthers our understanding that privacy cannot be guaranteed against such a server with only secure aggregation. Yet, in examples such as the message application discussed in Section 2, the described attack in this paper does *not* actually extends the capabilities of the server, as the federated learning system considered in the examples is implemented on a good-will basis without the ability for a user to inspect and vet the implementation of the protocol. From a security perspective, the server is so far not constrained to protocol, and is already capable of downloading user data - without federated learning. Both with and without our attack, the server in this example remains with current implementations only legally bounded to minimize their use of user data.

In terms of mitigation opportunities, we argue that strong user-level differential privacy, i.e. mechanisms applied directly on the user device to all outgoing model updates, remains the defense of choice against the attacks in the untrusted or malicious server threat model that we consider here.

REPRODUCIBILITY STATEMENT

We perform all our experiments based on the official implementation provided in Fowl et al. (2022). We provide details of parameters and evaluations for our experiments, as well as pre-processing steps for the *wikitext* dataset in Appendix A.

REFERENCES

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. *arXiv:1807.00459 [cs]*, August 2019. URL http://arxiv.org/abs/1807.00459.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing Federated Learning through an Adversarial Lens. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 634–643. PMLR, May 2019. URL https://proceedings.mlr.press/v97/bhagoji19a.html.

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 preprint arXiv:2112.02918*, 2021.

Kallista Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy Preserving Machine Learning. Technical Report 281, 2017. URL http://eprint.iacr.org/2017/281.

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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, December 2020. URL http://arxiv.org/abs/2005.14165.

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.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv:2010.11929 [cs]*, June 2021. URL http://arxiv.org/abs/2010.11929.

Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojtek Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted Transformers Breach Privacy in Federated Learning for Language Models. *arXiv:2201.12675 [cs]*, January 2022. URL http://arxiv.org/abs/2201.12675.

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.

Privacy Policy Google. How Messages improves suggestions with federated technology - Messages Help, 2022. URL https://support.google.com/messages/answer/9327902#zippy=%2Chow-your-data-is-protected%2Chow-federated-technology-works%2Cyoure-in-control.

Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering Private Text in Federated Learning of Language Models. *arxiv:2205.08514[cs]*, May 2022. doi: 10.48550/arXiv.2205.08514. URL http://arxiv.org/abs/2205.08514.

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.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold.

Nature, 596(7873):583–589, August 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Maximilian Lam, Gu-Yeon Wei, D. Brooks, V. Reddi, and M. Mitzenmacher. Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix. In *ICML*, 2021.

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

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)*, pp. 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.

Foundation OWASP. Deserialization of untrusted data, May 2022. URL https://owasp.org/www-community/vulnerabilities/Deserialization_of_untrusted_data.

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*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. *arXiv:2111.07380 [cs]*, December 2021. URL http://arxiv.org/abs/2111.07380.

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: Revisited and Enhanced. In Lynn Batten, Dong Seong Kim, Xuyun Zhang, and Gang Li (eds.), *Applications and Techniques in Information Security*, Communications in Computer and Information Science, pp. 100–110, Singapore, 2017. Springer. ISBN 978-981-10-5421-1. doi: 10.1007/978-981-10-5421-1_9.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. URL http://arxiv.org/abs/1706.03762.

Yuxin Wen, Jonas A. Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 23668–23684. PMLR, June 2022. URL https://proceedings.mlr.press/v162/wen22a.html.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019a.

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 (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14774–14784. Curran Associates, Inc., 2019b. URL http://papers.nips.cc/paper/9617-deep-leakage-from-gradients.pdf.

A Additional Experiment Details

In this section we provide additional details of our experiments. We start by detailing parameters. We set $\alpha = 10^{12}$ for GPT-2 and $\alpha = 10^{8}$ for BERT across all experiments. For GPT-2, we set $\beta = 10$ for GPT-2 if batch size is smaller than 256 and $\beta = 3$ for GPT-2 if otherwise. For BERT, we set $\beta = 10$ for all single keyword experiments, and we set $\beta = \frac{5}{K}$ for experiments on multiple keywords, where K is the number of keywords. We set d', the block size to be truncated and shifted as described in Section 3.2, to 32.

Next, we describe our procedure to process articles into sequences of tokens for each user. For experiments on GPT-2, the articles are tokenized with GPT-2 (BPE) tokenizer. For experiments on BERT, we tokenize the articles with the original BERT tokenizer. Then, given a sequence length ℓ, we concatenate all the words in an article into one array, and partition the array into sequences of length ℓ. The left-over words are discarded. Then, given the batch size B, we keep the first B sequences for each user and discard the rest. The user then computes their model update based on the remaining B sequences.

For each quantitative metric, the reported result is obtained over the average over the first 10 users with enough data. That is, for a given combination of batch size and sequence length, a user only contributes to the final result if his/her data is enough to fill in the required number of tokens.