Two Models are Better than One: FL for Next Word Prediction Is Not Private

Mohamed Suliman

Trinity College Dublin

Introduction

A notable real-world deployment of FL is within Google's Gboard, where FL is used to train the Next Word Prediction (NWP) model that provides the suggested next words that appear above the keyboard while typing. We present two attacks that reconstruct the original training data, i.e. the text typed by a user, from the FL parameter updates with a high degree of fidelity. We also show that adding Gaussian noise to the transmitted updates, which has been proposed to ensure local Differential Privacy (DP), provides little defence unless the noise levels used are so large that the utility of the model becomes substantially degraded.

Word Recovery Attack

In next word prediction the input to the RNN is echoed in it's output. The sign of the output loss gradient directly reveals information about the words typed by the user, which can then be recovered easily by inspection. This key observation is the basis of our word recovery attack.

word	i	$(\theta_1 - \theta_0)_i$
learning	7437	-0.0009951561
online	4904	-0.0009941629
is	209	-0.000997875
not	1808	-0.0009941144
so	26	-0.0009965639
private	6314	-0.0009951561

Table: Values of the final layer parameter difference at the indices of the typed words. Produced after training the model on the sentence "learning online is not so private", $E = 1, B = 1, \eta = 0.001$. These are the only indices where negative values occur.

To execute this attack in practice, simply subtract the final layer parameters of the current global model θ_0 from those of the resulting model trained on the client's local data, θ_1. The indices of the negative values reveal the typed words. Suppose the client's local data consists of just the one sentence "learning online is not so private". We then train model θ_0 on this sentence for 1 epoch, with a mini-batch size of 1, and SGD learning rate of 0.001 (FedSGD), and report the the values at the negative indices in Table 1.

Sentence Reconstruction Attack

We begin by selecting x_0 equal to the start of sentence token <S> and x_1 equal to the first word from our set of reconstructed words, then ask the model to generate $y_2 = Pr(x_2|x_0, x_1; \theta_1)$. We set all elements of y_2 that are not in the set of reconstructed words to zero, since we know that these were not part of the local training data, renormalise y_2 and then select the most likely next word as x_2. We now repeat this process for $y_3 = Pr(x_3|x_0, x_1, x_2; \theta_1)$, and so on, until a complete sentence has been generated. We then take the second word from our set of reconstructed words as x_1 and repeat to generate a second sentence, and so on. The Log-Perplexity of a sequence $x_0, ..., x_t$, is defined as

$$PP_\theta(x_0, ..., x_t) = \sum_{i=1}^{t} (-\log Pr(x_i|x_0, ..., x_{i-1}; \theta)),$$

and quantifies how 'surprised' the model is by the sequence. Those sentences that report a high perplexity for θ_0 but a comparatively lower one for θ_1 reveal themselves as having been part of the dataset used to train θ_1. Each generated sentence is scored by their percentage change in perplexity:

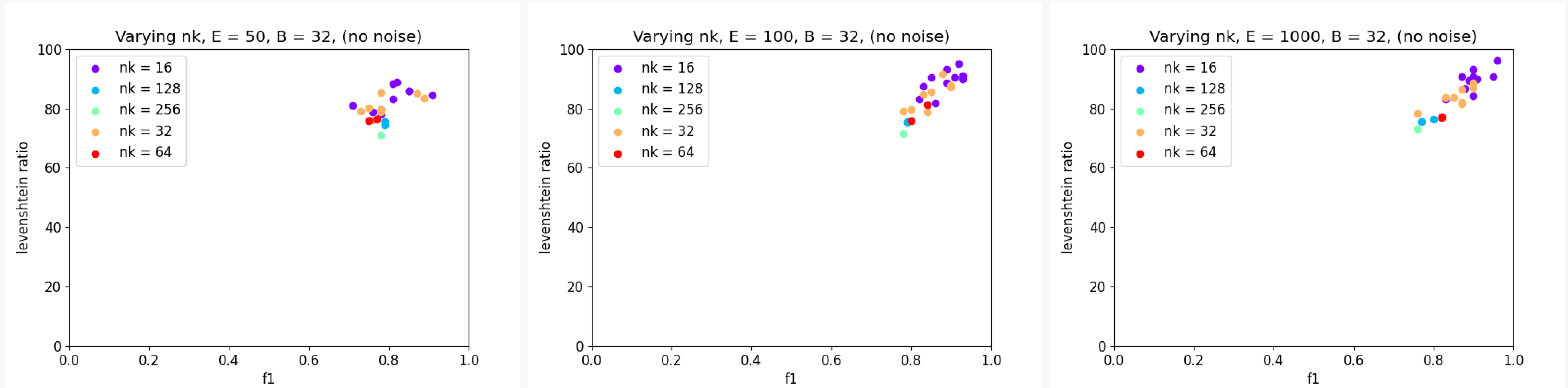
$$Score(x_0, ..., x_t) = \frac{PP_{\theta_0}(x_0, ..., x_t) - PP_{\theta_1}(x_0, ..., x_t)}{PP_{\theta_0}(x_0, ..., x_t)}.$$

By selecting the top-n ranked sentences, we select those most likely to have been present in the training dataset.

Possible Defences

The privacy situation may not be quite as bad; reconstructed text is effectively redacted due to the <UNK> token. Model dictionary choice thus plays a critical role. Character level language models also would likely make our attack much harder to perform.

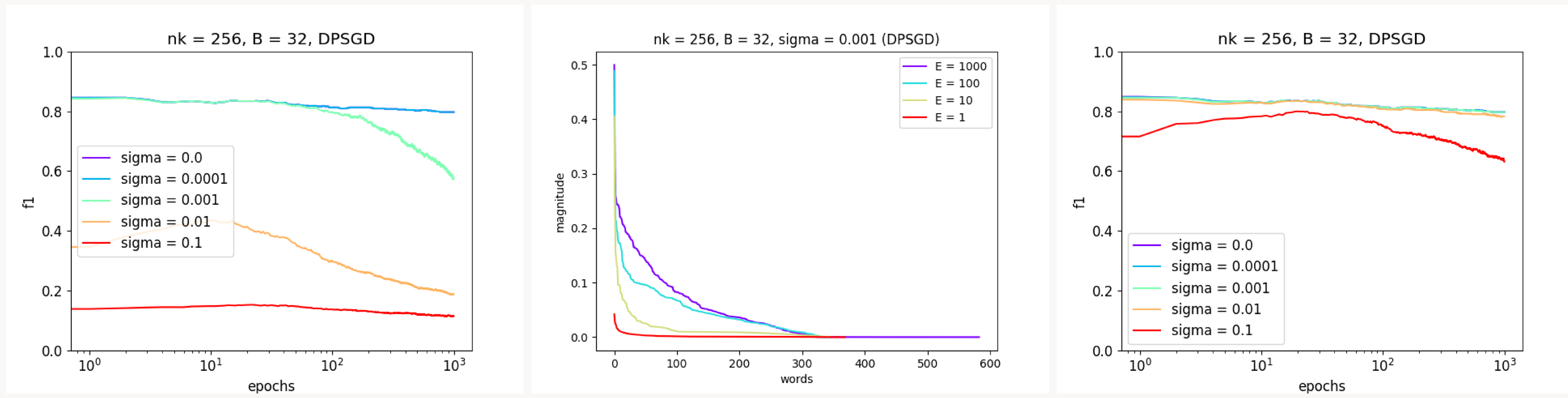
Performance against Vanilla FL



(a) $E = 50$ epochs (b) $E = 100$ epochs (c) $E = 1000$ epochs

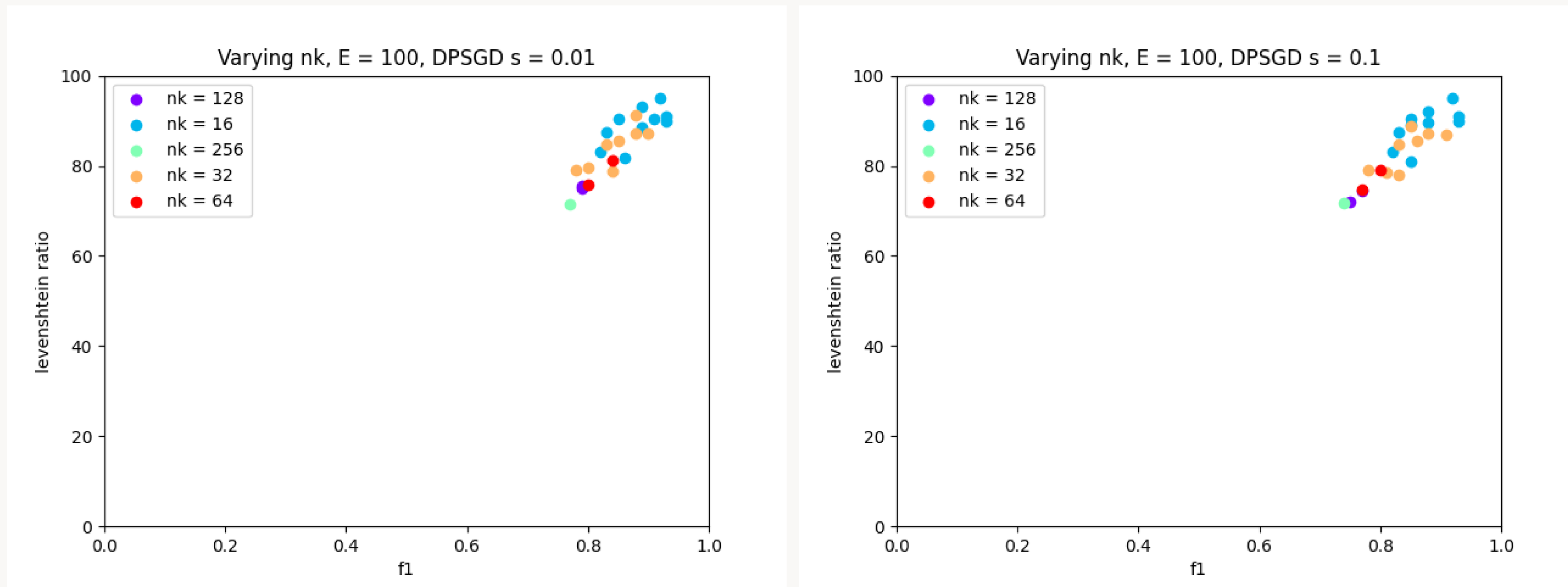
Figure: Reconstruction performance. Each point corresponds to a different dataset colour coded by it's size. The y-axis gives the average Levenshtein ratio of the reconstructed sentences. The x-axis is the F1 score between the tokens used in the reconstructed sentences and the ground truth. The closer a point is to the top-right corner, the closer the reconstruction is to perfect

Performance against FL with Local DP



(a) (b) (c)

Figure: Word recovery behaviour when Gaussian noise is all to local FL updates: (a) vanilla word recovery performance, (b) disparity of magnitudes between those words that were present in the dataset and those 'noisily' flipped negative, (c) word recovery performance when filtering is used.



(a) (b)

Figure: Sentence reconstruction performance with local DP for FedAveraging. show the reconstruction performance for different datasets colour coded by their size for DPSGD-like training, with $E = 100$, $B = 32$. Here we see no real effect in our results compared to the noise-free case.