The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks

Radek Bartyzal

GLAMI AI

24. 11. 2020

Motivation

Paper is by Google Brain, BAIR from 2019.

Current state:

- we are training large NLP models
- scraping a lot of data
- possibly confidential user data

Questions:

- can we extract e.g. card numbers (yes)
- is it due to overfitting? (no)
- how to quantify it? (exposure metric)
- Is my model likely to memorize and potentially expose rarely-occurring, sensitive sequences in training data?

Motivation

LONG LIVE THE REVOLUTION. OUR NEXT MEETING WILL BE AT| THE DOCKS AT MIDNIGHT ON JUNE 28 TAB
AHA, FOUND THEM!
WHEN YOU TRAIN PREDICTIVE MODELS ON INPUT FROM YOUR USERS, IT CAN LEAK INFORMATION IN UNEXPECTED WAYS.

Figure: There is an XKCD for everything [2].

Threat model

Threat model:

- black box attack
- 10 000s of queries
- sees logits / probabilities of the model outputs = it's harder without this

No Transformers?

They only test LSTMs and qRNNS not Transformers!

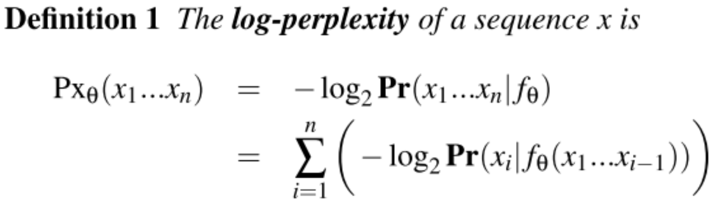
Methodics

Is my model likely to memorize and potentially expose rarely- occurring, sensitive sequences in training data?

Answer:

- insert randomly-chosen **canary** sequence into training data varying number of times
- how much models memorize = our **exposure metric**
- **exposure**: relative difference in perplexity between canaries and equivalent, non-inserted random sequences

Perplexity of a sequence



Definition 1 *The **log-perplexity** of a sequence x is*

$$\begin{aligned}
\mathrm{Px}_\theta(x_1 \ldots x_n) &= -\log_2 \mathbf{Pr}(x_1 \ldots x_n | f_\theta) \\
&= \sum_{i=1}^{n} \left(-\log_2 \mathbf{Pr}(x_i | f_\theta(x_1 \ldots x_{i-1})) \right)
\end{aligned}$$

Intermezzo:

- perplexity $= 2^{H(p,q)} \implies$ log-perplexity $= H(p,q)$
- Cross entropy $H(p,q) = -\sum_i^N p(x_i) log_2 q(x_i) \approx -\frac{1}{N} log_2 q(sequence)$
- for long sequences (Shannon-McMillan-Breiman theorem)

What are secrets?

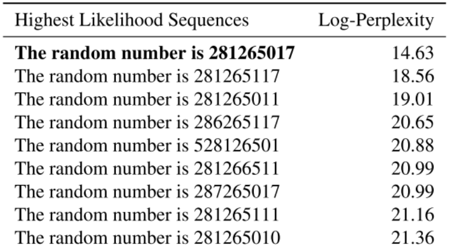
- NNs memorize some training data, thats ok if it helps to generalize
- Unintended Memorization = memorize useless data = secrets
- secret = represented by canary sequence
- canary = independent, random sequences from the input data
- $\implies$ canaries are useless for generalization
- $\implies$ insert canaries into training data
- $\implies$ evaluate their exposure in the trained model

Unintended Memorization

When trained neural networks may reveal the presence of out-of-distribution training data.

Exposure metric

- canary = sequence of 9 numbers not in training data
- candidates = other random sequences equal to canary = other 9 numbers that are not in training data
- exposure = $log(rank(canary))$
- $rank(canary)$ = position among candidates ranked by perplexity



Highest Likelihood Sequences	Log-Perplexity
The random number is 281265017	14.63
The random number is 281265117	18.56
The random number is 281265011	19.01
The random number is 286265117	20.65
The random number is 528126501	20.88
The random number is 281266511	20.99
The random number is 287265017	20.99
The random number is 281265111	21.16
The random number is 281265010	21.36

Estimating exposure = rank of canary

How to est. without calculating perplexity of all (10^9) candidates?

- sample some candidates
- fit skewed normal D over them
- calc. prob. of candidate perplexity $\leq$ canary perplexity

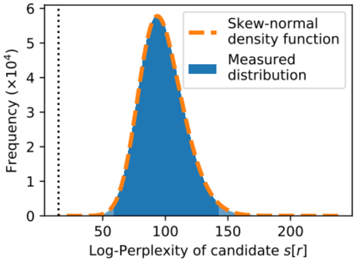


Figure: Skew normal fit to the measured perplexity distribution. The dotted line indicates the log-perplexity of the inserted canary, which is more likely (i.e., has lower perplexity) than any other candidate canary.

Experiments on small models

- 2-layer LSTM character-level
- PTB dataset
- single canary inserted = 9 digit random number

Memorization happens early in training

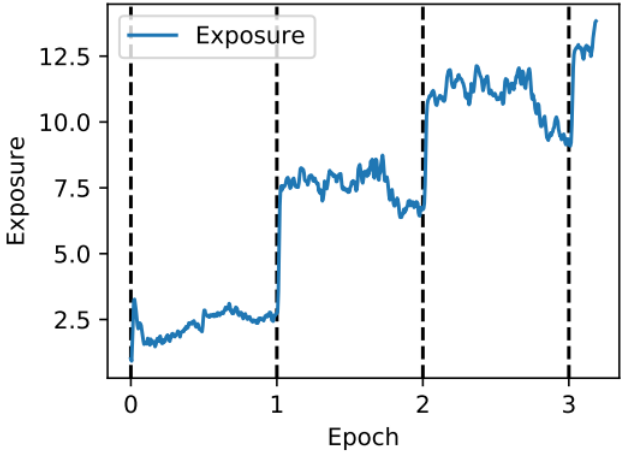


Figure: Exposure as a function of training time. The expo- sure spikes after the first mini-batch of each epoch (which contains the artificially inserted canary), and then falls overall during the mini-batches that do not contain it.

Memorization is not caused by overfitting

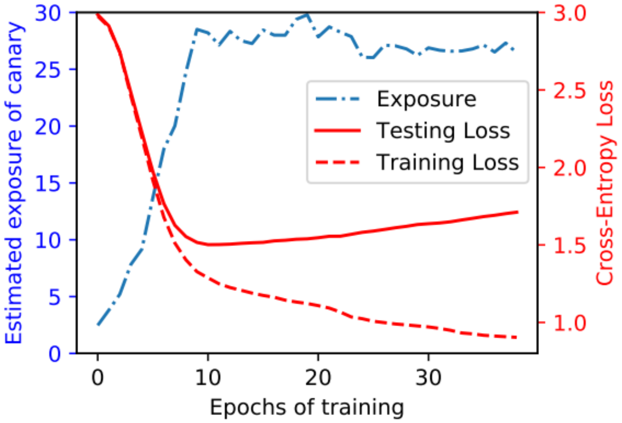


Figure: Comparing training and testing loss to exposure across epochs on 5% of the PTB dataset . Testing loss reaches a minimum at 10 epochs, after which the model begins to overfit (as seen by training loss continuing to decrease). Exposure also peaks at this point, and decreases afterwards.

High exposure implies extraction

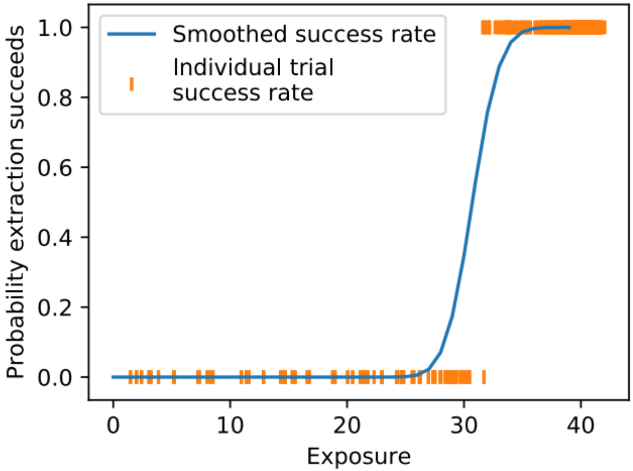


Figure: Extraction is possible when the exposure indicates it should be possible: when $|R = \textit{random space}| = 2^{30} \cong 10^9$, at an exposure of 30 extraction quickly shifts from impossible to possible.

How to extract

- the canaries had known fixed context at the beginning and end
- insert the starting context and see what the model predicts
- use shortest path search to get the most probable sequence

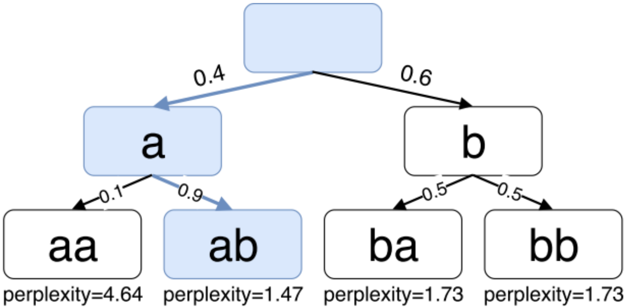


Figure: Shortest path search algorithm. Each node represents one partially generated string. Each edge denotes the conditional probability $Pr(x_i|x_1, \ldots, x_{i-1})$. The path to the leaf with minimum perplexity is highlighted, and the log-perplexity is depicted below each leaf node.

Experiments on large models

Memorization differs in models with same accuracy

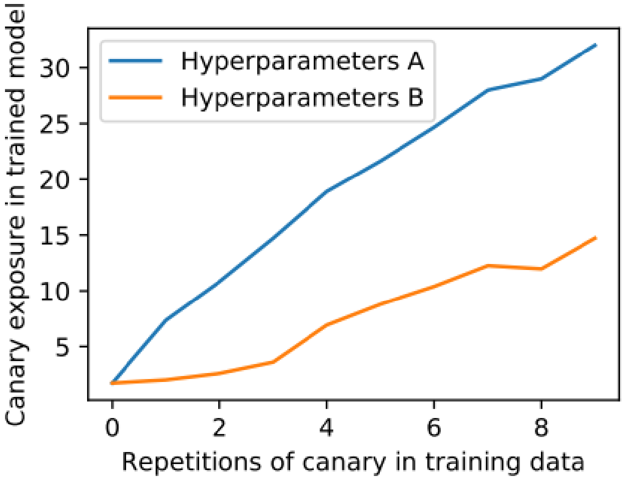


Figure: SOTA word-level language model trained to same accuracy with different hyperparams has very different exposure. If the canary occurs 9 times, it can be extracted from model A.

Smart Compose

- generative word-level
- trained on personal emails of millions of users
- commercially deployed for predicting sentence completion in emails
- current active use by millions of users
- predictions drawn not (only) from their own emails, but the emails of all the training users
- LSTM
- millions of parameters
- trained on billions of word sequences
- vocabulary size of tens of thousands of words
- canaries are 7 or 5 randomly selected words
- first and last two words are known context, and the middle 3 (or 1) words vary

Results

Smart Compose:

- does not sufficiently memorize canaries even after 1000s of insertions to training data

SOTA word-level on WikiText-103 (500MB):

- memorizes word canaries after 5-15 insertions

SOTA character-level on Penn-Tree-Bank (5MB):

- why not on WikiText again?
- memorizes numbers easily
- memorizes word canaries after 16 insertions but not enough to extract them

Differential Privacy

Differential Privacy (DP):

- adding 1 sample to training set does not significantly change model's output

DP-SGD [4]

1. calculate gradient of batch
2. clip gradient
3. add gaussian noise to the gradient

Training with DP-SGD removes the problem of memorizing secrets.

Sources

1. Carlini, Nicholas, et al. "The secret sharer: Evaluating and testing unintended memorization in neural networks." 28th USENIX Security Symposium (USENIX Security 19). 2019. https://arxiv.org/abs/1802.08232

2. XKCD https://xkcd.com/2169/

3. BAIR Blog post. https://bair.berkeley.edu/blog/2019/08/13/memorization/

4. Abadi, Martin, et al. "Deep learning with differential privacy." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016. https://arxiv.org/abs/1607.00133