Name: Extracting training data from Large language models

Paper link: https://arxiv.org/pdf/2012.07805.pdf

A. Introduction.

- Idea is to show that Large Language models that do not suffer from overfitting can also memorize training data. The LM can then be attacked to spit out this memorized training data
- Authors perform a type of **membership inference attack** that predicts whether a given string is in the training data.
- Use GPT2 as an example. It does not overfit because it trained on terabytes of de-duplicated data for only 12 epochs. Still, it learns examples in the training data.
- Authors make following contributions:
    - Provide methodology for generating candidate strings from LM which are then tested for presence in training data using **membership inference attack**
    - Black box access method that is generalizable to any large LM
    - Provide a clear example of how such attacks are practical and hence need serious consideration.
    - Show that an increase in LM size leads to an increase in memorization.
    - Authors were successfully able to extract personal information, emails, UUIDS from the LM.

B. Description

- Some extent of memorization is expected from the LM by the virtue of the learning objective it is optimized on + overfitting. The authors are not talking about such a type of memorization.
- **Model Knowledge Extraction:** A string **s** is extractable from a LM if given a prefix, the LM at hand assigns high probability to it.
- **k-Eidetic memorization:** A string s is said to be memorized if it can be extracted from the LM and it is found in <=k training examples.
- The attack process is composed of:
    - **Text generation**: Allow LM to generate candidate strings by 3 possible strategies
        - W1: Autoregressive text generation by inputting <start> token as prefix
        - W2: Sampling with temperature decay to encourage more exploration by LM.
        - W3: Conditioning on internet text by sampling possible prefixes from the internet and then using it to get strings from LM. [BEST]
    - **Filter candidates**: The idea here is to keep only those candidates that have been **assigned high probability by LM and are rare strings**.
        - W1: Perplexity : Lower the better.

        - W2: Likelihood of string s by GPT-2 large / Likelihood to string s by GPT-2 small or medium: Higher the better. [BEST]
        - W3: Perplexity of lowercase data.
    - **Remove duplicates.**
    - **Check for membership:** Manually searched on Google.
- The authors find that W3 of text generation combined with W2 of filtering the candidates is the best way of coming up with candidate strings that are **RARE SEQUENCES THAT ARE MEMORIZED** (exactly what authors wanted to target.)

C. My assessment/ Analysis

a. Easy to understand attack process.
b. Authors portray that such attacks are PRACTICAL especially for large LM.
c. The paper is very well written providing a good overall idea.

D. Limitations:

a. Very limited analysis on the value of k in k-eidetic memorization.
b. Unable to understand whether the unit of training example in the paper and GPT2 is the same e.g. document or string etc.
c. I am not very sure whether the actual situation is as alarming as the authors portray it to be.