# Deep USC Interview: Data Condensation for Text with GPT-2

**Shoumik Atul Gandre**
Department of Computer Science
University of Southrn California
Los Angeles, CA 90089
shoumika@usc.edu

## Abstract

The perplexity of a GPT-2 model is computed on train, validation and test split of the WikiText-2 dataset as a baseline perplexity. We propose a method to perform Gradient Matching algorithm on discreet text tokens and produce a synthetic token dataset of size (120, 128).

## 1 Baseline GPT-2 Language Model on WikiText-2 Dataset

### 1.1 Training

The Baseline GPT-2 model is trained on the train split of the WikiText-2 dataset. The preprocessing of the data involves the following steps. First, filter out any sentence with length less than 1. Second, tokenize the sentence. Third, truncate any sentence with sentence length greater than 128 tokens to 128 tokens. Finally, pad any sentence with length less than 128 tokens to 128 tokens. After preprocessing the data, train GPT-2 on the preprocessing training split for 10 epochs with a batch size of 16 using cross entropy loss.

### 1.2 Perplexity

Perplexity of a sentence $\mathbf{S} = (s_1, s_2, ..., s_T)$ is computed with the following formula

$$PPL(\mathbf{S}) = \exp\left\{-\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(s_t \mid s_{<t})\right\}$$

The perplexity of a split is computed as the mean of the perplexities of all the sentences in the split. Therefore, the perplexity for a split $\mathbf{X} = (S_1, S_2, ..., S_N)$ of size N, the perplexity is computed as

$$\text{mean-PPL}(\mathbf{X}) = \frac{1}{N}\sum_{n=1}^{N} PPL(S_n)$$

Since, our maximum sentence length is always 128 as described in the problem statement, T=128. The perplexities on train, validation and test split of a GPT-2 model trained on the train split of WikiText-2 data are present in Table 1

## 2 Data Distillation

### 2.1 Analysis on Compression

WikiText-2 (real dataset) after preprocessing, has sequences of length 128 and there are over 40000 unique tokens. If we were to have a distilled dataset of shape $\mathbb{R}^{128\times 120}$, that implies that The meaning

Table 1: Perplexity of GPT-2 trained on real WikiText-2 data

| Split | Perplexity |
|---|---|
| train | 12.6559 |
| validation | 36.3179 |
| test | 30.5602 |

of sequentially ordered 40000 tokens need to be successfully represented by sequentially ordered 15,360 tokens. This could be possible if a token can capture the meaning of atleast 3 other tokens for the distilled dataset. Tokens that could be used interchangeably (like synonyms) can be helpful for this task.

## 2.2 Representing discreet text tokens as differentiable tensors

Images are represented as continuous 2 dimensional Matrix with 3 channels $(Height \times Width \times Channels)$. However, Text data is represented by label encoding the words with an id. For example, a sequence can be: (1, 3, 4, 2, ...., 2). We can see that if we represent images in a continuous tensor, we can acquire it's gradients. However, with our current representation of Text, we cannot acquire the gradients for our sentences. Instead, we need to think about our sentences as a sequentially ordered set of probability distribution of tokens. For example, (1, 3, 4, 2, ...., 2) can be represented as a one-hot-encoded representation:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & ... \\ 0 & 0 & 1 & 0 & ... \\ 1 & 0 & 0 & 1 & ... \\ 0 & 1 & 0 & 0 & ... \\ ... \\ 0 & 1 & 0 & 0 & ... \end{pmatrix}$$

Note that, now we need to use the embeddings layer as a matrix multiplication for each token. For example,

```
embeddings = nn.Embeddings(50257, 768)
synthetic_tokens = torch.rand((120, 128, 50257), requires_grad=True)
```

The embeddings are computed using:

```
em = synthetic_tokens @ embeddings.weight
```

We can represent a sentence in the matrix $\{0,1\}^{N \times |T| \times |V|}$ where $N$ is the number of sentences, therefore for distilled data, N is 120, $|T|$ is the fixed size of sentence which is 128 and $|V|$ is the size of vocabulary, which is the default 50257 of GPT-2 Tokenizer. Therefore, for our distilled data can be represented in the matrix $\{0,1\}^{120 \times 128 \times 50257}$. Now, this representation can be freely used to compute the gradients. Although, by the end of Gradient Matching techniques, we will end up with a matrix $\mathbb{R}^{120 \times 128 \times 50257}$ Therefore, to revert it back to tokens and a dataset of size (120, 128) we can use an argmax function on the last axis. However, this causes a loss in data. Therefore, this is not an ideal method to be used for gradient matching.

## 2.3 Solution

We shall be using Gradient Matching algorithm [2] to produce synthetic embeddings. Furthermore, we shall be incorporating ideas from [1] in order to apply gradient matching for text. First, we pass Real Data through GPT-2, compute the loss and compute the gradients with respect to all model parameters. Second, we pass synthetic tokens through GPT-2's wpe layer and obtain synthetic embeddings. We shall copy and detach these embeddings. Third, we pass the detached embeddings through the remaining GPT-2 layers. Fourth, we compute the match_loss from Gradient Matching paper and then Update the embeddings. Fifth, we decode the updated embeddings back to tokens. Sixth, we pass these new synthetic tokens through the network and repeat.

## References

[1] Ilia Sucholutsky and Matthias Schonlau. “Soft-Label Dataset Distillation and Text Dataset Distillation”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021. DOI: `10.1109/ijcnn52387.2021.9533769`. URL: `https://doi.org/10.1109%2Fijcnn52387.2021.9533769`.

[2] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. “Dataset Condensation with Gradient Matching”. In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=mSAKhLYLSsl`.