Text Dataset Condensation Investigate

Enyu Zhao
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
enyuzhao@usc.edu

Abstract

This article is aimed to take investigate text dataset condensation and test the performance on a non-pretained GPT-2 Model.

1 Preparing the GPT-2 Model

GPT-2 is a large transformer-based language model with 1.5 billion parameters and aims to predict the next word, given all of the previous words within some text. In this specific task, we are required to train a GPT-2 model from scratch using WikiText-2 as dataset for baseline benchmark. We used GPT2LMHeadModel provided by hugging face as the GPT-2 model to use.

In NLP(Natural Language Processing) pipeline, we need to first turn the text into numbers that the model can process, the numbers known as tokens. For different models, there are different tokenizers tailored for them to gain optimal performance. So we use the GPT-2 tokenizer for training our GPT-2 model.

2 Metric

One of the metrics that evaluates language models' performance is perplexity. It measures how well a language model can predict a sequence of words or a text sample. In essence, perplexity quantifies how surprised a language model would be when encountering a new unseen sequence of words. A lower perplexity indicates that the model is more confident and accurate in predicting the next word, while a higher perplexity suggests more uncertainty and lower predictive performance.

Perplexity is calculated based on the probability distribution of the model's predictions. The model assigns probabilities to different words or sequences of words, and perplexity measures how well these probabilities match the actual data. The mathematical definition can be defined as Eq.1:

$$PP = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, \ldots, w_{i-1})\right) \quad (1)$$

Here, w_i means the $i-th$ word in the sentence. In our experiment, however, we use the cross-entropy with the predicted sentence without the last word and the original sentence without the first word to act as $-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i|w_1, w_2, \ldots, w_{i-1})$

3 Dataset Condensation Methods Tested

With large-scale models becoming more and more dominant these days, in order to reach state-of-art performance, training the model requires an enormous dataset and extremely long training time. To

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

solve this problem, dataset condensation (Zhao, et al. 2021) was proposed. It can be used to condense a large dataset into a small set with synthetic samples without sacrificing too much performance. The author later proposed Dataset Condensation with Distribution Matching as the previously mentioned Dataset Condensation has the problem which is synthesizing the dataset is still computationally expensive as complex bi-level optimization and second-order derivative computation are involved.

This method synthesizes condensed datasets by matching feature distributions of the synthetic and original dataset in sampled embedding spaces. It significantly reduces the synthesis cost while achieving comparable or better performance. Due to the reduction of the synthesis cost, it can be scaled to much larger datasets with more complicated neural networks. The author formulates the synthesizing process as a distribution matching problem such that the synthetic data are optimized to match the original data distribution in a family of embedding spaces by using the maximum mean discrepancy measurement.

By assuming that each training sample $x \in \mathbb{R}^d$ can be embedded into a lower dimensional space by using a family of parametric functions, they use the empirical estimate of the maximum mean discrepancy and turn the goal of synthesizing dataset into the following problem:

$$\min_{\mathcal{S}} E_{\vartheta \sim P_\vartheta, \omega \sim \Omega} || \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \psi_\vartheta(\mathcal{A}(x_i, \omega)) - \frac{1}{|\mathcal{S}|} \sum_{j=1}^{|\mathcal{S}|} \psi_\vartheta(\mathcal{A}(s_j, \omega)) ||^2 \quad (2)$$

Here, ψ_ϑ is the embedding function parametrized by ϑ, x_i is the data sample in original dataset and s_j is the data sample in the synthesized dataset. The whole training process by learning the synthetic data $\mathcal{S}$ by minimizing the discrepancy between two distributions in various embedding spaces by sampling ϑ. It requires only optimizing $\mathcal{S}$ but no model parameters like θ and thus avoids expensive bi-level optimization.

A quick idea is to transfer this condensation algorithm which is originally developed for the image dataset into the text dataset. However, there are certain problems.

3.1 Datatype of the Synthetic Dataset

Problem As we need to optimize the synthetic dataset $\mathcal{S}$ by applying this algorithm, we need the dataset $\mathcal{S}$ to be floating number because we want to use Stochastic Gradient Descent as the optimizer, thus we need the dataset to have gradients. However, the dataset is now text data and is comprised of the ids of the tokens that build the sentence. This difference means if we perform the gradient descent to optimize the synthetic dataset $\mathcal{S}$, we will end up with the dataset filled with floating numbers instead of integers representing the ids of the tokens. Normally when we use the GPT-2 language model, we are forced to feed the model with the ids, and the optimized synthetic dataset won't meet up with this requirement as the elements inside are floating numbers.

Our Solution With limited time, we only tried the intuitive method which is forcing datatype transformation. We argued that forcing the datatype to be integer before feeding the dataset into the model, we may lose a certain amount of accuracy and therefore a better transformation method can be our future focus.

Also we have to deal with the numbers that are out of bounds. Here the bounds are the vocabulary of the tokenizer applied, so if the synthetic dataset has element that is not in the numerical range of the tokenizer's ids, the GPT-2 model just won't start training as it fails to embed the dataset. Again, we tried the intuitive method, which is clipping the dataset to force the element to fall into the tokenizer's vocabulary.

3.2 The Embedding Function

This problem is more like doing a choice between different embedding functions ψ_ϑ, we implemented two different embedding functions: MLP and pretrained GPT-2. We think this embedding function choice is a problem because those two embedding functions all have its own drawbacks.

MLP With MLP setting to be the embedding function, we have the problem listed in 3.1 which is the data type of the final optimal synthetic dataset $\mathcal{S}$ will be floating number and cause problems.

GPT-2 With GPT-2 setting to be the embedding function, we have other problems.

While using the GPT model to embed the synthetic dataset does avoid the annoying synthetic dataset's problem of translating the elements' datatype, the GPT-2 model's requirement for interger input just bring the datatype translation problem to a different place which is before the embedding.

The other problem is GPT-2 model is way larger than the MLP we created thus we don't enough time and computation power to tune or getting any further understanding.

3.3 Our Method Conclusion

We randomly choose 120 sentence from the original dataset and set them to be the initialized synthetic dataset. We then applied the embedding process and calculted the maximum mean discrepancy using MLP and pretrained GPT-2 as different embedding functions ψ_ϑ. After gained the optimal synthetic dataset $\mathcal{S}_{MLP}$, we tested GPT-2 model on this dataset with clipping and naive datatype transformation applied. The final perplexity is 56.67 while the training on the whole dataset gained the perplexity of 126.67. However, the model trained on the whole dataset used different hyper-parameters and may not achieve the optimal performance. By searching online, we found the optimal performance of GPT-2 in such situation will gain the perplexity of around 29.7.

We would claim this condensation method, though not perfect or delicate, still has great potential and achieved milestone performance. However, we hadn't got the chance to test $\mathcal{S}_{GPT}$ provided by using GPT as embedding function due to limited time.

4 Unimplemented Ideas

Due to the limited time, we have some other ideas that may contribute to the condensation method but remain untested.

For example, when evaluating the distance of the embedded tensors, could we use other methods other than maximum mean measurement? Cosine Similarity can be powerful in such a situation. And we never tried coreset selection methods which can avoid the datatype problems, they may bring a better solution to this problem as no accuracy is abandoned.