Natural Language Understanding report
Fine tuning GPT2

Samuel Kostadinov (232315)

University of Trento

samuel.kostadinov@studenti.unitn.it

This document is the report written for the MS.C. course "Natural Language Understanding" of the University of Trento.
The goal of the project described in this report was fine-tuning GPT2 on three different domains.

1. Introduction

Fine tuning is a process that uses a the weights of a pre-trained neural network as starting point to train the neural network on a specific domain. In this case, being GPT2 a general language model, the fine tuning aims to making this model more domain-specific. To do so, usually the model is trained for a small number of epochs on a dataset created based on the domain someone is interested into. Doing so, the model will perform better on the domain it is fine-tuned for.

2. Task Formalisation

The project's goal is to use the Huggingface's version of GPT2 (either the small or the medium one) and fine tune it on at least three different domains. The domains were left to be chosen by the students. The students then should:

- Find the datasets to use in the fine-tuning process, belonging to three different domains.
- Present the dataset's statistics
- Create a baseline for the model
- Fine tune and evaluate the model

The fine tuning is a part of transfer learning, and, as stated in [1], there are different ways to perform this operation. One of the possible ways to fine-tune a model is to keep the architecture and train it using the already trained weights as starting point. With this method, the most obvious choice for the baseline is to use the non-fine tuned model. In this way, the evaluation can be done on the same model and it's easier to understand what is the actual improvement of the model after the fine tuning process.

3. Data Description & Analysis

The dataset chosen for fine tuning GPT2 in this case are the following:

- **ArXiv**:
  This dataset has data about some papers, published between 2010 and 2020, about quantum physics. The collected data contain the abstract, which is the field used in this project, the title, the categories, an id, the DOI and a field called "created".
- **IMDB movie reviews**:
  This dataset was originally thought for movie review sentiment analysis. Even though it's not a language modelling specific dataset, since it contains the text of the review, it can be adapted to this goal.
- **wiki_movie_plot**:
  This dataset contains data, scraped from Wikipedia, about movies. These data account, among others, Title, Director, Cast, Genre and Plot. Since the plot is one of the data scraped, it's possible to fine tune a language model with this dataset.
- **wikitext-2**:
  This dataset contains data about various Wikipedia articles without a specific topic.

All these datasets are taken from kaggle.com. The ArXiv, the IMDB and the wiki_movie_plot are in the form of csv files, while the wikitext is not. The statistics of the datasets are listed in the following.

- The ArXiv dataset has 72881 examples in total, with a vocabulary size of 124,837. These examples are then divided into train and test sets. In particular, the train set has a total number of 7,104,092 tokens and an average sequence length of 142.281. The test set, on the other hand, has 3,510,281 tokens with an average sequence length of 152.969. The number of OOV tokens is 28,530.
- The IMDB dataset has 50,000 examples in total, with a vocabulary size of 194,770. These examples are then divided into train and test sets. In particular, the train set has a total number of 13,945,306 tokens and an average sequence length of 279.465. The test set, on the other hand, has 28,801 tokens with an average sequence length of 288.010. The number of OOV tokens is 183.
- The Wiki_plot dataset has 35,000 examples in total, with a vocabulary size of 182,448. These examples are then divided into train and test sets. In particular, the train set has a total number of 14,989,004 tokens and an average sequence length of 429.657. The test set, on the other hand, has 8,153,791 tokens with an average sequence length of 410.936. The number of OOV tokens is 0.
- The WikiText-2 dataset has 41,076 examples in total, with a vocabulary size of 33,278. These examples are then divided into train and test sets. In particular, the train set has a total number of 2,088,628 tokens and an average sequence length of 57.159. The test set, on the other hand, has 245,569 tokens with an average sequence length of 56.780. The number of OOV tokens is 6384.

The different domains involved in this projects are: quantum physics, reviews, movie plots and informative text

4. Model

For this project, since the goal is fine tuning, the model was already implemented. Huggingface gives the opportunity to use different implementations of GPT2, with different sizes in particular. For this project, the

goal was to fine tune either the small or the medium version of GPT2. The complete specs are at https://huggingface.co/transformers/v2.2.0/pretrained_models.html but the ones of the small and medium model are reported also here.

- Small model: This model has 12 layers for a total number of 117 million parameters.
- Medium model: This model has 24 layers and a total number of 345 million parameters.

For this project in particular, due to CUDA Out Of Memory related issues, I used the small model. The project, in my case, is written in two different scripts, one for the fine tuning procedure and one for the evaluation procedure.
The evaluation procedure measures the perplexity (PPL), and is strongly influences by an already existing script that Huggingface used as example in https://huggingface.co/docs/transformers/perplexity. This script first loads the data, then gets rid of the fields of the csv that are not needed. After that the dataset discards all the data used for the training and keeps only the test set. The test set is then tokenized with the pre-trained GPT2 tokenizer. After tokenizing the dataset, the model is tested. The loss function is saved and then used to compute the perplexity.
The training procedure is different. Although the differences are present, there are some similarities. In fact, as in the evaluation, the script first reads the data and then gets rid of the fields that are not useful for the fine tuning. For the fine tuning itself, a pytorch dataset is required. Using the idea from https://colab.research.google.com/drive/13dZVYEOMhXhkXWfvSMVM1TTtUDrT6Aeh?usp=sharing#scrollTo=0NmMdkZO8R6q, there is a class that extends the pytorch dataset and using the DataLoader class the data are divided into batches. Then there is a pytorch train loop that fine tunes the model, saving it at the end of every epoch as checkpoint and at the end of the whole loop.
In both the evaluation script and the fine tuning script the data are tokenized, and to do so I used the pre-trained tokenizer for GPT2 by Huggingface.
For this project in particular, the baseline used is the non-fine tuned version of GPT2, since it could give a good measure on how much the model improved after fine tuning.

5. Evaluation

The main metric used in this project, to evaluate the fine-tuned model, is perplexity, abbreviated as PPL. Perplexity is defined as the exp of the cross-entropy of the language model on the test data given a distribution p generating the language L and the attempt to model it q. This means that the language model has to be tested on some test data and using the cross-entropy loss as basis, the perplexity is computed.
It's worth noticing that there are two versions of the wikitext dataset in the table, because it was tested with two possible values for the max_length parameter. The two values were 1024 (the default value provided by Huggingface) and 100, which is the value used for the other datasets, due to CUDA Out Of Memory issues.

Dataset	Non fine tuned (mean)	fine tuned (mean)
ArXiv	$9.418006 \cdot 10^{15}$	$1.727806 \cdot 10^{15}$
IMDB dataset	20,821,760,000,000	8,626,600,000,000
wiki_plots	4,416,732,000	534,096,000
WikiText-2(1024)	25.1705	25.1705
WikiText-2(100)	45.0797	45.0797

For the ArXiv dataset, the standard deviation of the PPL was $1.0806294638666854 \cdot 10^{16}$ if non fine tuned, while it decreased to 1717865415691811.5 after fine tuning.
For the IMDB dataset, the non fine tuned version had a standard deviation of the PPL of 7670249819138.879, while it was 5652541326253.175 after the fine tuning process. The wiki plots with the non fine tuned model, had a standard deviation equal to 5151889774.477905, and after fine tuning GPT2, the standard deviation of the PPL became 496,416,104.37212855
On the wikitext dataset, both the non fine tuned and the fine tuned version had a standard deviation of 0.0. This happened with both the tested versions, which means that happened with both the version with the max_length parameter set to 100 and the version with the same parameter set to 1024.

In the table are there is the report of the results of the experiments. The fine tuning had a predefined number of 5 epochs of training, and a predefined batch size of 2. The batch size is smaller than expected, since it is usually higher, but a higher batch size would result in CUDA Out Of Memory, so due to hardware limitation the choice is to use a smaller batch size. Regarding learning rate and epsilon, I experimented a few values, and the ones that gave the best results were $5 \cdot 10^{-4}$ as learning rate and 10^{-8} as epsilon. The other values that were tested for epsilon were: 0, 10^{-10}, 10^{-9}, 10^{-7}. The other values tested for the learning rate were: 10^{-3}, $5 \cdot 10^{-3}$, 10^{-4}, $5 \cdot 10^{-4}$, 10^{-5}, $5 \cdot 10^{-5}$.

In the table that lists the results of the evaluations performed before and after fine tuning, it's possible to see that the fine tuning operation leads to better results in the perplexity of the models. Due to the hardware limitations of the components used, the results were not as good as they could have been. In fact, event testing the scripts on Google Colab, some hyper parameters could not be set to a high value. For example, in the evaluation script, the max_length parameter should have been set to 768 or 1024, while on this project was set to 100 due to the physical limitations. To prove that this parameter would have made the difference, the smallest dataset used, the WikiText-2, was tested with both values. The PPL resulting from the test with 1024 is 55% of the PPL resulting from the evaluation with 100.
Another consideration that is worth noticing is that the PPL for the WikiText-2 dataset is the same before and after the fine tuning procedure. This most likely happened because GPT2 is a very big model and already achieves a very good result on that dataset. In fact, even without fine tuning the loss was around 0.2, a value that the fine tuning process was not able to improve very much, and this lead to the fact that the PPL is the same. On bigger and more specific datasets, though, the fine tuning process shows a difference in the computed PPL.

6. Conclusion

This project, although with some limitations, improved the performance of the model.
The major problems were connected with hardware limitations, but even with this problem there were some interesting results. In some cases, with very specific and big datasets, the fine tun-

ing behaved like expected, showing an improvement, while on the other hand, a smeller and not-so-specific dataset proved that the starting model is actually an excellent starting point.

References

[1] Dishashree Gupta. *Transfer learning and the art of using Pre-trained Models in Deep Learning*. 2017. URL: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/ (visited on 06/07/2022).