GPT-2 fine-tuning and evaluation

Luigi Riz - Mat. 223823
University of Trento
Via Sommarive, 9, 38123 Povo, Trento, TN
`luigi.riz@studenti.unitn.it`

Abstract

This work focuses on the language modelling task of fine-tuning the well known causal transformer GPT-2, proposed by Radford *et al* [1]. In particular, this report describes the peculiarity of this model, the different datasets used to train it and provides a description of the results achieved. The whole work is based on Hugging Face[2], a Python library to manage language models, datasets, text generation pipelines and much more. As a development environment, Google Colab is used, in a way to exploit the computational power it offers and work totally on cloud.

1 GPT-2: introduction

OpenAI GPT-2 is a huge transformer-based language model with 1.5 billion parameters that was trained on an 8 million web page dataset, named WebText. Due to this large amount of data it was fed with, and thanks to its complex structure, it can be used for different tasks, such as language modelling, sequence classification or token classification. According to the paper used to present it, the model was able to reproduce state-of-the-art results on 7 out of 8 tested language modelling datasets without any fine-tuning, showing the great capabilities of such a kind of model. Moreover, the authors underline that, despite the huge amount of parameters the model is made of, it still doesn't overfit the training dataset, proving the effectiveness of the use of transformers for the aforementioned types of tasks.
In 2020 OpenAI released a new version of the model, called GPT-3[3], containing 174 billion parameters and trained with different datasets, such as Common Craw (570 GB after filtering), WebText, the English version of Wikipedia, Books1 and Books2. As it can be understood easily looking at the paper, the performances of this new model are far better than the ones of GPT-2 and represent the state-of-the-art regarding transformer-based language models.

1.1 Hugging Face Transformers

Hugging Face provides a simple library to experiment with transformers for free[4]. In particular it provides several pretrained models, such as BERT, RoBERTa, GPT and GPT-2, that can be easily downloaded and tested even locally. Moreover, to reduce the computational requirements when dealing with research projects, it provides reduced versions of the above models: for example, it is possible to instantiate a small version of GPT-2 (`gpt2`), a medium one (`gpt2-medium`), a large one (`gpt2-large`) or the complete one (`gpt2-xl`), by simply inserting these tags when calling the load function. Moreover, the library is perfectly integrated with PyTorch and Keras, providing an efficent framework for training and testing the models.

2 Datasets

To evaluate GPT-2 performances and fine-tune it, three different datasets have been collected. In particular, the Hugging Face Dataset library[5] has been used, since it provides very simple ways to download, preprocess and manage textual data. The used datasets relate to different domains, namely Wikipedia articles, Amazon reviews and news articles coming from news sites.

2.1 WikiText

The WikiText corpus is a dataset containing 100 million words extracted from a set of articles of the English Wikipedia. It has been collected and introduced by Merity *et al.*[6] and is presented as an alternative to the Penn Treebank dataset, that was considered too small to learn models that capture long-term dependencies. It comes in two different versions, namely `WikiText-2` (the one used in the project) and `WikiText-103`, that differ for the size of the training set. Both the datasets come already divided into three splits (training, test, validation), each of them is a list of dictionaries. The only key inside each list item is `"text"`, whose related value is a string representing a paragraph inside the article.

	WikiText-2			WikiText-103		
	Training	Validation	Test	Training	Validation	Test
Articles	600	60	60	28.475	60	60
Tokens	2.088.628	217.646	245.569	103.227.021	217.646	245.569

Table 1: Statistics of the WikiText datasets.

Table 1 summarises the statistics of the two datasets. The authors underline that the only difference between the two datasets consists in the training dataset, which is much larger in the `WikiTest-103`. Oppositely, the exact same articles compose the validation and the test set for both the versions. The only consequence of these decisions is visible in the size of the vocabularies: `WikiText-2` has a much smaller vocabulary when compared to `WikiTest-103`'s one.

2.2 Amazon reviews

The Multilingual Amazon Reviews Corpus (MARC)[7], is a large dataset providing reviews data for multilingual text classification. It contains reviews in multiple languages, *i.e.* English, Japanese, German, French, Spanish, and Chinese, and it contains multiple fields that are discarded in this project, since they are not needed. In fact, each record is a dictionary containing the following keys: `"language"`, `"product_category"`, `"product_id"`, `"review_body"`, `"review_id"`, `"review_title"`, `"reviewer_id"`, `"stars"`. The only information retained during preprocessing is the `"review_body"` field of the English split of the dataset, that is renamed `"text"` for the sake of uniformity.
The available statistics of the processed dataset are represented in Table 2.

2.3 CC_News

CC_News is a dataset containing news articles from news sites all over the world. It has been created thanks to news-please[8], a web crawler and information extractor for news, developed by Hamborg *et al.* The version proposed by Hugging Face contains 708.241 articles, that represent only a small portion of English language subset of the original CC-News dataset. Perhaps it is not natively split into training, test, and validation set, so this passage is needed during preprocessing. For the goal of the project, most of the fields of the records (`"date"`, `"description"`, `"domain"`, `"image_url"`, `"text"`, `"title"`, `"url"`) are discarded, retaining only the `"text"` component. Moreover only 7083 articles are used, otherwise the fine-tuning phase would exceed the two hours of runtime (note that a news article contains far more tokens than a review and it is much heavier to manipulate). As a result, 6374 articles are deputed to fine-tuning, whereas 709 are dedicated to evaluation.

3 Dataset preprocessing

As suggested by the Hugging Face documentation in a notebook about language modeling[9], the datasets are subject to some preprocessing before being used for fine-tuning.
In particular, as a first step, each record of the dataset is passed through the tokenizer, regardless of the length of the text. Doing so, some warnings about the length of the chunks are raised by the system, but they can be ignored, since the problem is solved with next step. At this point, each record in the dataset is represented as a sequence of tokens, each of them linked to a word. The last step of preprocessing consists in concatenating all the tokens in the dataset and successively divide them into uniform-length chunks. In this way all the samples will need the same space in memory and the fine-tuning will proceed smoothly even with low memory and GPU resources.
As a final remark, it is worth underlining that the two preprocessing steps can be applied using the built-in function `datasets.map(...)`, that will exploit parallel computation over several batches. This results in a faster preprocessing of the datasets provided in input.

4 Fine-tuning GPT-2

The fine-tuning of the model is carried out using the Trainer API provided by HuggingFace[10]. In particular, the optimizer used is Adam with fixed weight decay, the one proposed by default by the library. Only some parameters are modified from their default value, namely the weight decay, which is set to 0.01, and the learning rate, which is set to 2×10^{-5}. The whole process is carried out over three epochs, with a total processing time of about an hour and a half for each dataset. The choice of such a small number of epochs is guided by the need not to exceed the runtime limit provided by Google Colab, especially when working with GPUs.
At the end of each epoch, the framework reports the training and the validation loss, permitting to monitor how the computation is proceeding. It is interesting to note that the biggest drop in the loss, and consequently in the perplexity, is observable between the

	MARC		
	Training	Validation	Test
Reviews	200.000	5.000	5.000

Table 2: Statistics of the Multilingual Amazon Reviews Corpus (MARC) dataset.

first and the second epochs, whereas the third epoch tends to improve the metric only by small quantities. This suggests that further increasing the number of epochs will cause only small improvements in performance, even leading to some overfitting of the model on the dataset.
Note that the models are saved locally once the fine-tuning ends. Doing so, we avoid to run every time the training procedure, that would result in a waste of computational resources and time.

5 Evaluation and results

The evaluation of the fine-tuning procedure is performed in two phases. In the first one, each model is analysed by itself, comparing the perplexity of the original GPT-2 model with the perplexity of the fine-tuned architecture. Moreover, before and after training, the samples with lowest and highest perplexity are extracted. Then the two couples are compared, to observe the changes in perplexity and eventually highlight some other modifications in the behaviour of the model. As a second phase, the models are compared one with the others, evaluating both the perplexity when applying to new domains (*e.g.* by applying the WikiText fine-tuned model to the Amazon Reviews dataset) and analysing their generative ability, by providing different sentence starts and comparing the outputs generated by the models.

5.1 Single-model analysis

As explained above, the first phase of the evaluation of the fine-tuning procedure is performed analysing a model at a time. In particular two aspects are observed: the evolution of the training loss (and consequently of the perplexity over the validation set) and the changes in the samples with highest and lowest loss.

5.1.1 Entropy loss and perplexity

Thanks to the Hugging Face framework, the evolution of the loss during the training phase is easily observable and is reported in Table 3 for each of the three datasets. Note that `Epoch 0` refers to the loss measured on the validation set using the original GPT-2 model.

As highlighted in the table, the biggest improvements in terms of loss are observable with the fist epoch of training, whereas the other two epochs are useful only to adjust the model. Another aspect to point out is the low validation loss the original GPT-2 model demonstrates on the CC_news dataset. This behaviour suggests that the GPT-2 model is somehow used to process textual data in the form of news articles, maybe because the data it was trained on is in a news-articles-like form.
Besides the entropy loss, also perplexity is evaluated, following the approach described in the article by Ravi Charan[11]. In particular, formula (1) is applied

$$Perplexity(M) = e^{H(L,M)} \tag{1}$$

where $H(L, M)$ is the cross entropy for model M on language L. In this particular case, the entropy loss is the output of the evaluation of GPT-2 using the validation set. Consequently, to extract the perplexity of the model on a given dataset before and after fine-tuning, the only thing to do is to pass through the exponentiation of the returned loss. As a reference, Table 4 collects the perplexity values for each dataset, before and after the training procedure.

5.1.2 Positive and negative samples

Besides the use of the metrics described above, also an empirical analysis is performed. In fact, for each dataset, a positive and a negative sample (*i.e.* the chunk with highest perplexity/loss and the one with the lowest) are collected from the validation set. Then, after fine-tuning, the same procedure is applied, resulting in the collection of two pairs of chunks for each dataset. At this point the couples are matched and compared, to understand how the perplexity values change with fine-tuning and if the chunks extracted in the two phases are the same or if they are different. Table 5 summarizes the collected evidences.
This empirical analysis produces some noteworthy facts. The easiest to note is the high improvement in perplexity for the negative samples, that is bigger than the drop observable for positive samples. Moreover, it is interesting to underline that the selected chunks before and after fine-tuning are the same in 4 cases out of 6. These two facts suggest that the model is adapting well to the new domain, being able to generalize properly also on the evaluation dataset.

	WikiText		MARC		CC_news	
Epoch	Training loss	Validation loss	Training loss	Validation loss	Training loss	Validation loss
0	-	4.0796	-	4.0910	-	3.6291
1	3.4877	3.4106	3.7505	3.6471	3.4371	3.3071
2	3.3615	3.3958	3.6548	3.6162	3.3382	3.2954
3	3.3016	3.3930	3.6079	3.6098	3.2808	3.2932

Table 3: Evolution of the entropy loss in the fine-tuning procedure.

	Loss		Perplexity	
	Before fine-tuning	After fine-tuning	Before fine-tuning	After fine-tuning
WikiText	4.08	3.39	59.12	29.76
MARC	4.09	3.61	59.80	36.96
CC_news	3.63	3.29	37.68	26.93

Table 4: Entropy loss and perplexity values before and after fine-tuning.

5.2 Cross-model analysis

The second phase of the evaluation of fine-tuned models is performed by comparing their performances when dealing with the same task. As a first step, all the models are evaluated on all the datasets, in a way to highlight similar behaviours by different models. As a second phase, the same start of sentence is provided to all the models, in a way to be able to compare their generative capability.

5.2.1 Perplexity

As explained above, in the first phase of the cross-model analysis we consider the perplexity of the models when dealing with the datasets presented in this paper. In particular, for each dataset and for each model (including the original GPT-2 for completeness), an evaluation step is performed, obtaining as output the perplexity values for each combination. Results are represented in Table 6.
This analysis highlights an interesting aspect in fine-tuning. In fact, looking at the results, it is clear that the best performances on a single datasets are achieved by the models fine-tuned on it (as expected). However, when looking at the results of a model over the three datasets, then the best results are achieved by the original GPT-2 (together with the CC_news GPT-2). This suggests that fine-tuning is a trade-off between generalization capability and specificity on a given context: fine-tuning definitely boosts the performances on the target domain, but it also reduces the ability of the model to fit datasets different from the training one.
The good overall performances of CC_news GPT-2 have a simple explanation: since the original GPT-2 model has already good perplexity values on the CC_news dataset, then the fine-tuning of it (that produces the CC_news GPT-2) does not change too much the weights of the models. In this way, the fine-tuned model has a boost in performance on the target domain, but it still retains good generalization capabilities, that allow for decent perplexity values also on other datasets.

5.2.2 Generative capability

As a second step of the cross-model analysis, some sentence starts are provided to each of the four models, in a way for them to produce the rest of the phrase. Once the outputs are produced, they are empirically evaluated, trying to focus on the generative capability of the models. In particular, four different sentence starts are provided: the first one is very general, whereas the other three are extracted from the validation sets of the three datasets, in a way to observe how different fine-tunings affect the behaviour of the models. The generated sentences are available in the Colab Notebooks.
The results of this empirical analysis confirm the considerations exposed in Section 5.2.1. In fact, also in this case the original GPT-2 is the model with highest generalization capabilities, whereas the fine-tuned models stand out when dealing with data similar to the one they have been trained on, but a suffer a lot when working in new contexts.

6 Possible improvements

Some approaches can be explored to further improve the effectiveness of fine-tuning. In particular it would be interesting to work on the full-size GPT-2 model, which should be less prone to overfitting and should have higher generalization capabilities. Moreover, some different methods of training could be explored, to refine the way the models are fine-tuned. Lastly, new datasets should be used, obtaining models specific for unexplored domains such as the financial, the movies and the healthcare ones.

	Positive Sample			Negative Sample		
	Before fine-tuning	After fine-tuning	Same chunk?	Before fine-tuning	After fine-tuning	Same chunk?
WikiText	14.07	6.58	NO	279.58	129.82	YES
MARC	13.76	10.52	YES	224.35	153.86	YES
CC_news	2.05	1.33	YES	338.36	305.99	NO

Table 5: Perplexity of the positive and negative samples before and after fine-tuning. Column `Same chunk?` specifies if the negative(positive) chunk extracted before fine-tuning is the same as the one extracted after the training procedure. See the Colab Notebook for further information.

	Original GPT-2	WikiText GPT-2	Amazon Reviews GPT-2	CC_news GPT-2
WikiText	59.1217	**29.7551**	90.9659	67.1725
MARC	59.8010	131.4979	**36.9601**	58.6074
CC_news	38.8794	94.4016	55.7375	**24.1946**

Table 6: Perplexity of the models presented in the paper on all available datasets.

References

[1] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[2] Hugging face - the ai community building the future. `https://huggingface.co/`. Accessed: 2021-10-13.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[4] Hugging face - transformers. `https://huggingface.co/transformers/index.html`. Accessed: 2021-10-13.

[5] Hugging face - datasets. `https://huggingface.co/docs/datasets/`. Accessed: 2021-10-13.

[6] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016.

[7] Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. The multilingual amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[8] Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March 2017.

[9] Hugging face - language modeling. `https://github.com/huggingface/notebooks/blob/master/examples/language_modeling.ipynb`. Accessed: 2021-10-13.

[10] Hugging face - trainer. `https://huggingface.co/transformers/master/main_classes/trainer.html`. Accessed: 2021-10-13.

[11] The relationship between perplexity and entropy in nlp. `https://towardsdatascience.com/the-relationship-between-perplexity-and-entropy-in-`
Accessed: 2021-10-13.