Introduction to Machine Learning

A. Bernacchioni, E. Fuochi, N. Signani

June 2025

Abstract

The aim of this project is to develop a language model that generates short stories for children following some narrative guidelines given in the form a subset of six fixed tags. We decided to fine-tune the DistilGPT2 model on the TinyStories dataset exploiting the LoRA (Low Rank Adaptation) approach designed by Hu et al. (2021)[1]. We describe how we pre-processed the dataset, the model architecture and the training procedure we followed, before presenting the results obtained in terms of cross-entropy loss and evaluation of other metrics on the test set.

Architecture and approach. We decided to base our model on the DistilGPT2 transformer available in the *Hugging Face transformers* library[4], as some of its stated potential uses include the generation of creative fictional texts and literary art. However, we looked for an approach that would lower the hardware barrier for fine-tuning, making it more efficient from a storage and computational point of view. That is why we decided to exploit the LoRA (Low Rank Adaptation) for LLMs presented by Hu et al. (2021)[1][3]. The strategy consists of freezing the pre-trained model weights except the attention layers of the transformer that are optimized with rank decomposition matrices of the updates to those weights. Thus, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, we constrain the update to be of the form $W_0 + \Delta W = W_0 + \frac{\alpha}{r}BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, α a scaling factor to control the strength of the adaptation (which we set to $\alpha = 2r$), and the rank $r \ll \min(d,k)$. The theoretical principle behind this strategy comes from the observation that when adapting to a specific task, pre-trained models perform well and learn efficiently despite projections to a smaller subspace.

Tokenizer. We used Byte-Pair Encoding (BPE)[2] tokenization. BPE creates a base vocabulary of tokens by taking all the symbols used in the dataset. Then, until a desired vocabulary size is reached, it adds new tokens by merging those already present in the vocabulary so far. A new token obtained this way will be added to the vocabulary if it is the most frequent pair of existing tokens.

Prepocessing. The TinyStories dataset contains short children stories associated with zero to six tags corresponding to some of their narrative qualities: *BadEnding*, *Conflict*, *Dialogue*, *Foreshadowing*, *MoralValue*, *Twist*. These stories were selected further according to the following criteria.

In order to select the best stories to train the model, we decided to introduce a *Quality score*, which rewards stories with good vocabulary diversity, sentence variety, and containing direct speech and penalizes stories excessively long or short, stories containing bad patterns (such as repetitions, very long words, too many special characters), and stories with poor vocabulary. We excluded stories with Quality score lower than 0.7, and stories that don't have tags associated to them, as they wouldn't give the model information on how to generate according to certain tags. Finally, we sampled 10000 stories satisfying our quality criteria from the full training dataset, in order to train our models on the best instances. Since a few elements of this dataset are not of an appropriate length because they have not beeen penalized enough by the Quality score, the final training dataset is obtained by removing those stories and has a final size of 8464 samples. 1000 of this were used as validation dataset. We report in Table 1 the distribution of tags across the final dataset.

Dialogue	*Twist*	*Foreshadowing*	*BadEnding*	*MoralValue*	*Conflict*
4972	2619	1753	1670	1608	1418

Table 1: Distribution of tags in the training dataset. The majority of *Dialogue* tags is due to the presence of dialogue in most stories.

We standardized the format of the stories by removing overly repeated blank spaces and empty lines. Then, the text has been saved in a dictionary containing features and story. As we did not want to predict the tags, we trained the model only to predict tokens in the story and not to predict the labels, being given a prompt containing the features.

Training. Since Google Colab allows for limited time of free T4 GPU computing, we decided to run the training algorithm in local on our machine, using a NVIDIA GeForce MX150 GPU.

The standard choice for an optimizer when using the resources made available by *Hugging Face* for LoRA is AdamW. We also used a warmup scheduler in the first phase of training. During this part, the learning rate increases until it reaches a standard value that we fixed at 2×10^{-4}. Then, a standard decay scheduler of the learning rate takes over, as displayed in Figure 1. The warmup phase makes up 10% of the training time. Training operations were performed in batch, with a batch size of `batch_size` $= 16$. We experimented with 3 different values of the rank r of the LoRA decomposition, $r = 8, 16, 32$, training the models for 5 epochs. In Table 2 we present a summary of the number of trainable parameters and training time of the models.

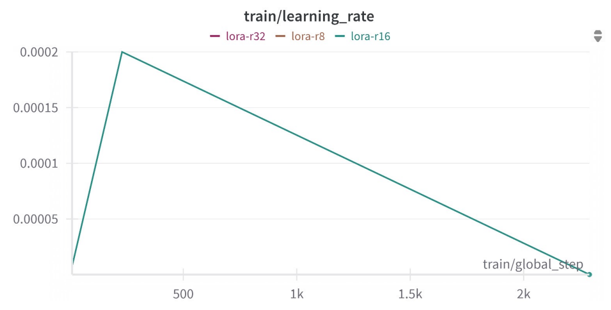


Figure 1: Evolution of the learning rate during training.

Rank	**Trainable parameters**	**Trainable parameters** (%)	**Training time**
8	$405,504$	0.4926%	7h 3m 17s
16	$811,008$	0.9804%	8h 1m 57s
32	$1,622,016$	1.9417%	9h 37m 14s

Table 2: In the third column we highlight the reduction in number of trainable parameters with respect to the base model DistilGPT2, which has 83,534,592 parameters.

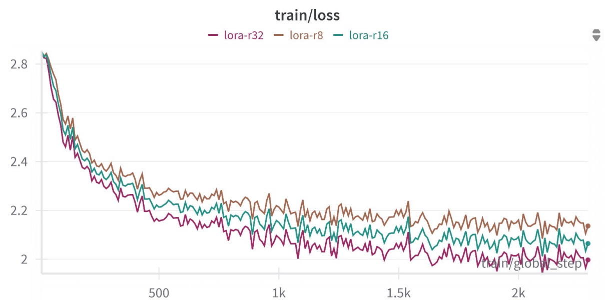
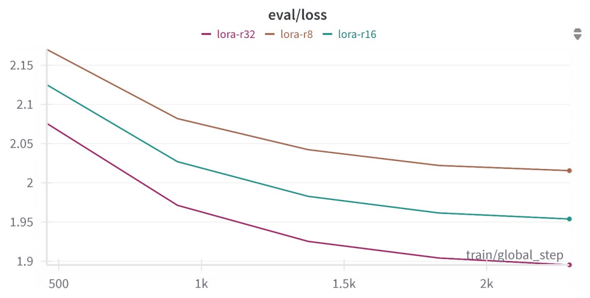


Figure 2: Training loss for LoRA ranks $r = 8, 16, 32$.



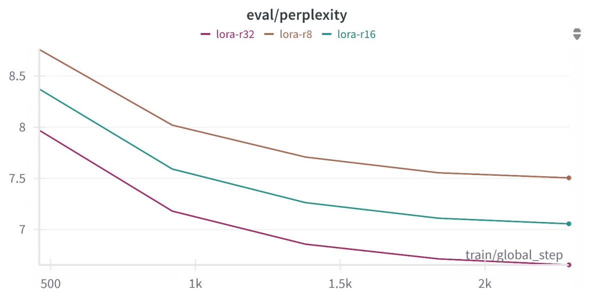


Figure 3: Validation loss and perplexity for LoRA ranks $r = 8, 16, 32$.

Story inference e metrics. Using the three trained models, we generate stories prompting the models with a string containing the desired features, corresponding exactly to the format on which they were trained. We used the *Hugging Face* `generate()` method, setting its parameter `do_sample=True`, so as to choose the following token during generation based on the probability distribution found during fine-tuning. Moreover, this allowed us to experiment with the `temperature` parameter, used to incentivize an exploratory behaviour that results in more creative stories.

In order to evaluate the quality of our models, we considered the following metrics.

- **Average cross entropy loss and perplexity.** It measures the confidence of the model in the prediction of the next token.
- **Quality score.** It's the same score used to preprocess the dataset. It indicates the quality of a story according to the criteria chosen as described above. Higher values correspond to higher frequency of "good patterns" in the story. For reference, the Quality score computed on the original test dataset is $QS = 0.8638$.
- **Self-BLEU.** It measures diversity in a set of generated texts, showing how similar each sentence is to the other. Ranging from 0 to 1, with 0.3 indicating a good model, high values indicate lower diversity, i.e. generated texts are similar to each other. We computed it by taking a smoothed mean of the n-gram overlap for $n = 1, 2, 3, 4$ bewteen a candidate story and all other stories used as reference. We reported the average Self-BLEU score computed for all stories in the test set.
- **Novelty.** It measures how much the generated stories differ from the stories in the training set. We obtained it computing the average self-BLEU score with a generated story as candidate and 100 sampled stories from the training dataset.

	Base model	**LoRA rank 8**	**LoRA rank 16**	**LoRA rank 32**
Average cross-entropy loss	2.7245	2.0177	1.8425	1.8972
Perplexity	15.5399	7.8727	6.6797	7.0295

Table 3: Comparison of the results for the models with different LoRA ranks and the base model.

Finally, in Table 3, we compare the results of the average per-token cross entropy loss and perplexity obtained with the models fine-tuned with rank $r = 8, 16, 32$ and of the base model without tuning. The best performance was obtained with $r = 16$. Indeed, it is a good compromise between low ranks, which require less computation, and higher ranks, which risk overfitting on the dataset used for fine-tuning. However, the effectiveness of fine-tuning with rank as low as $r = 8$ is evident, as it achieves a perplexity much lower than the base model. We also report in Table 4 how the models performed when choosing different values of temperature $T = 0.6, 0.8, 1.2$. Higher temperatures make for better metrics, however, since our metrics are unable to quantify story coherence, they don't necessarily generate better stories. Indeed, the models tend to become more creative, but in doing so they also generate stories which are less coherent and align less to the required prompted tags.

Rank	**Temperature**	**Self-BLEU**	**Novelty**	**Quality Score**
8	0.6	0.501	0.793	0.468
	0.8	0.240	0.765	0.572
	1.2	0.106	0.826	0.644
16	0.6	0.366	0.565	0.732
	0.8	0.290	0.625	0.712
	1.2	0.111	0.824	0.732
32	0.6	0.354	0.592	0.624
	0.8	0.280	0.637	0.732
	1.2	0.127	0.810	0.748

Table 4: Comparison of the metrics when tuning the temperature.

Upon these final considerations, we decided to set our model with LoRA rank $r = 16$ and temperature $T = 0.8$, which is a good compromise between creativity and coherence of the generated stories.

Contributions. The group actively collaborated on the project. In particular, Nicolò focused on the training procedure and the implementation of LoRA, Andrea and Enrico focused on the choice of the metrics and evaluation of the model and results.

References

[1] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*, 2021. Available at arXiv:2106.09685.

[2] Documntation available at Hugging Face: DistilGPT2

[3] Documentation available at Hugging Face: LoRA

[4] Documentation available at Hugging Face: BPE