Yxir Report 04-25 🤖

Mattéo Serrano

Introduction

- This experiments occurs on the `dataset_big_patent_v3.json`
- We decided to conduct this experiment to train a LLM to learn and generate technical content on different subjects

Goals & Scientific Challenges (& Sota Analysis)

- The goals of this experiment were to fine-tune an LLM to do inference on a Q&A dataset + context.
- We measure success on this experiment by using a traditional human evaluation
- On this experiment, we assume that on the given big patent dataset, that :
  - The "anchor" column was like the context
  - The "query" seems to be obviously the question
  - The "positive" seems to be the answer to the question
- The hypothesis we wanted to test is that with fine-tuning, training a LLM on a specific dataset allows to theoratically - although it was not proven in this experiment - to reach or go beyond the SOTA especially on specialized tasks like patent for exemple
- The experiment was supposed to test the robustness of zero-shot and especially the progress that can do fine-tuning on the inference of a small-sized LLM
- The SOTA for zero-shot for very large LLM seems to be `GritLM/GritLM-8x7B` and for a good performance with a small LLM `e5-small-v2` seems to be a good compromise and for fine-tuning the is `bge-m3` that I wanted to use but is too big or `intfloat/multilingual-e5-large-instruct`

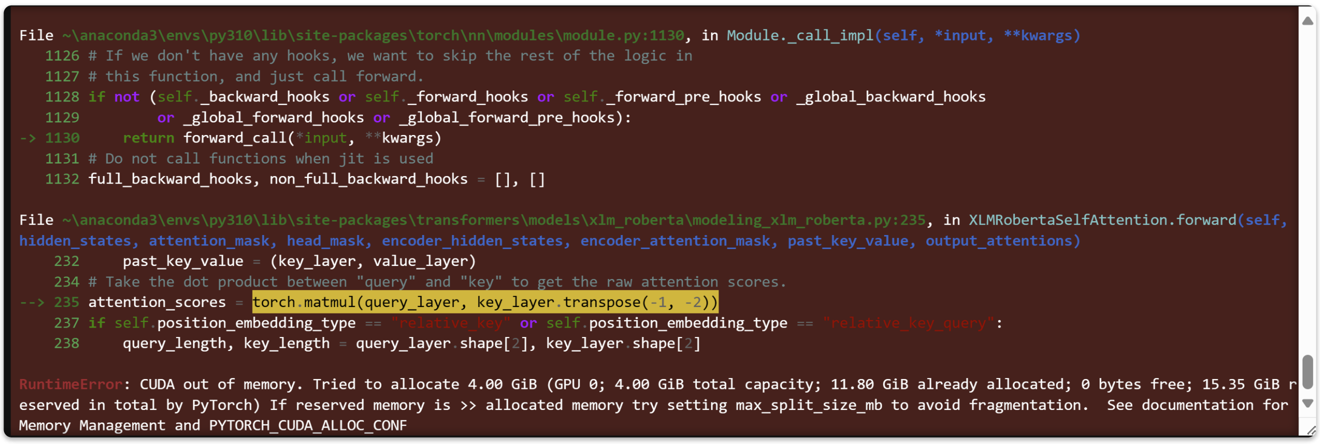
Contribution

1. We used fine-tuning of a LLM and more precisely, like it is said in this article, we used supervised learning to fine-tune the model and instruction training

which aims to improve model performance in answering questions or responding to user prompts with context for example.

The steps in details is that first we chose a quantified LLM to fit in my small 4 Go of Vram on my GPU.
Then we formated the dataset in order to have a text in a prompt that can be tokenized.
After this, we defined the TrainingArguments
Then we did a zero-shot performance evaluation
Then we fine-tuned the model with LoRa because of the fact that the LLM is quantified and we can't directly fine-tune all the weights after that we train the model

- We choose such techniques like the Parameter Efficient Fine-tuning with LoRA because in practice, the authors of the LoRA paper cited a 10,000x reduction in parameter checkpoint size using LoRA fine-tune GPT-3 compared to full parameter tuning
- The limitation of our contribution with fine-tuning a LLM lies in the small dataset (499 rows) that I suspect is prone to overfitting, the clear limitation of my equipment that allowed me only to take the smallest LLM possible and I had a lot of trouble during the training phase :



```
File ~\anaconda3\envs\py310\lib\site-packages\torch\nn\modules\module.py:1130, in Module._call_impl(self, *input, **kwargs)
   1126 # If we don't have any hooks, we want to skip the rest of the logic in
   1127 # this function, and just call forward.
   1128 if not (self._backward_hooks or self._forward_hooks or self._forward_pre_hooks or _global_backward_hooks
   1129         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1130     return forward_call(*input, **kwargs)
   1131 # Do not call functions when jit is used
   1132 full_backward_hooks, non_full_backward_hooks = [], []

File ~\anaconda3\envs\py310\lib\site-packages\transformers\models\xlm_roberta\modeling_xlm_roberta.py:235, in XLMRobertaSelfAttention.forward(self, hidden_states, attention_mask, head_mask, encoder_hidden_states, encoder_attention_mask, past_key_value, output_attentions)
    232     past_key_value = (key_layer, value_layer)
    234 # Take the dot product between "query" and "key" to get the raw attention scores.
--> 235 attention_scores = torch.matmul(query_layer, key_layer.transpose(-1, -2))
    237 if self.position_embedding_type == "relative_key" or self.position_embedding_type == "relative_key_query":
    238     query_length, key_length = query_layer.shape[2], key_layer.shape[2]

RuntimeError: CUDA out of memory. Tried to allocate 4.00 GiB (GPU 0; 4.00 GiB total capacity; 11.80 GiB already allocated; 0 bytes free; 15.35 GiB r
eserved in total by PyTorch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid fragmentation.  See documentation for
Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

Results

1. Zero-shot :

> *Question: How does the crowdsourcing method is used to adjust a video game element ?*
> *Answer: The crowdsourced method uses the data of other users and then creates a new version of the game. This can be done in several ways, including creating a*

> *new level that will have more interesting gameplay elements, or by adding new levels to an existing one*
>
> *Expected Answer : A processor retrieves a plurality of received game element feedback data from a plurality of users of a game and causes the game element to be adjusted during execution of the game*

Human evaluation : Pretty good but not perfect

2. Fine-tuned LLM

TO DO

The results confirm our hypothesis that fine-tuned LLM is better than zero shot performance (Wow, who would have thought ????)

Examples

None

Knowledge

- Fine-tuning is hard and a science in its own right
- It takes time (a lot)
- Librairies dependencies suck (but I already knew that).
- Fine-tuning > zero-shot performance
- A fine-tuned LLM even small can outperform large generalized LLM on specific tasks

What have I done wrong ?
Hopefully not a lot of things but I don't know that because if I knew, these errors would not exists

Properly ?
I have fine-tuned an LLM for text generation with a Q&A dataset for the first time with little prior knowledge and a lot of research

Conclusion and next steps

In conclusion, with this project we saw how we can fine-tune a LLM to a specific topic like patent. In doing so, it can lead to a LLM that is more performant than a more generalistic one. We also compared the performance of zero-shot and fine-tuning.

One thing that I did not do and will be my next step, is to play with hyperparameters and find the best ones but I did not have the time, like learning rate, batch size, number of epochs, etc. Maybe I can also improve zero-shot performance by doing prompt engineering but I still have a lot of research to do in this domain.

Github repository :
https://github.com/Matt504/Yxir_04_25_Synthetic_big_patent_Similarity