Project 5: LLM

SW13 – Jannine Meier – FS24

Preprocessing: Winogrande dataset

Dataset used: winogrande_l

1. Define a Prompt template

```
"""Below is an instruction that describes a task, paired with an input that provides further context. Write a response that contains only one digit either 1 or 2 and no other words or symbols.

### Instruction:
{}

### Input:
{}

### Response:
{}"""
```

Preprocessing: Winogrande dataset

2. Define an instruction for all data samples

Instruction = "Choose which option is the right one to replace the underscore and make the sentence meaningful and answer with a number (1 or 2) only"

3. Define the inputs by concatenating the sentence and both options

Input =	Sentence with underscore	+ option1	+ option2
Input =	"... to create space. The _ is small.""	+ couch	+ garage

4. Define the response by taking the correct answer label

Response =	Correct answer label (1 or 2)
Response =	2 (only for train_dataset)

Preprocessing: Winogrande dataset

Stemming and Lemmatizing

Remove punctuation and stopwords

6. Tokenize the sequence using the model's tokenizer

-> Byte-Pair Encoding (BPE) technique

-> add special tokens (BOS and EOS), add padding to max_seq_length (PAD = EOS)

7. Full decoded formatted example

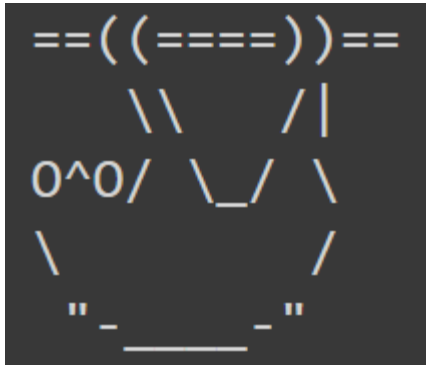
<|beginning_of_text|>
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that contains only one digit either 1 or 2 and no other words or symbols.

Instruction:
Choose which option is the right one to replace the underscore and make the sentence meaningful and answer with a number (1 or 2) only

Input:
Ian volunteered to eat Dennis's menudo after already having a bowl because _ despised eating intestine. Option 1: Ian Option 2: Dennis

Response:
2<|end_of_text|>... <|end_of_text|>

Model: Pretrained LLM



llama-3-8b-bnb-4bit

Developed by Unsloth using the latest Meta Llama 3 architecture

Model Specification

- Pre-trained Knowledge
  - great language understanding as the base model is trained on 15 trillion tokens
  - 8 billion parameters
- Quantization
  - use a 4-bit quantization to reduce memory usage
    - can be finetuned up to 2.4 times faster and requires 58% less memory compared to standard configurations
- Language and Training
  - model is trained on a diverse corpus covering multiple languages, but it is primarily optimized for English text generation

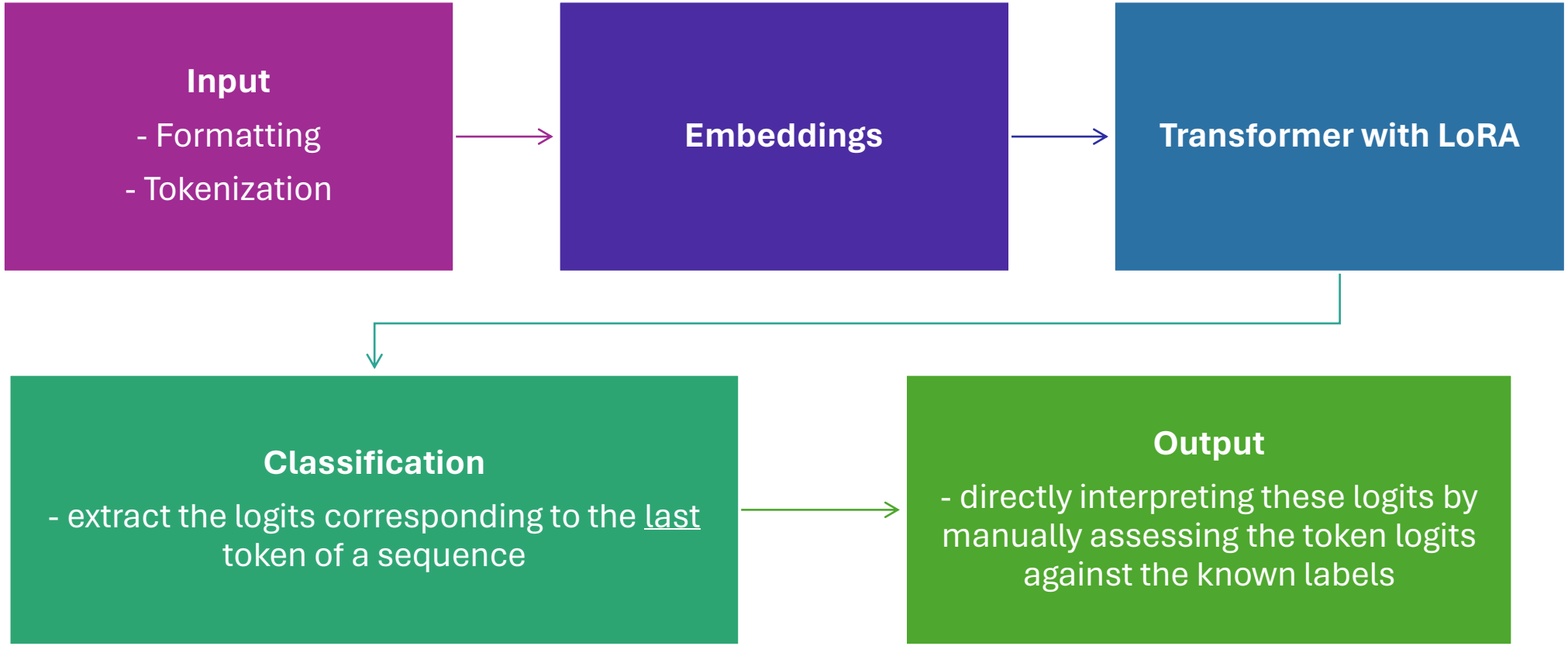
LoRA (Low-Rank Adaptation) Adapters

Configurations

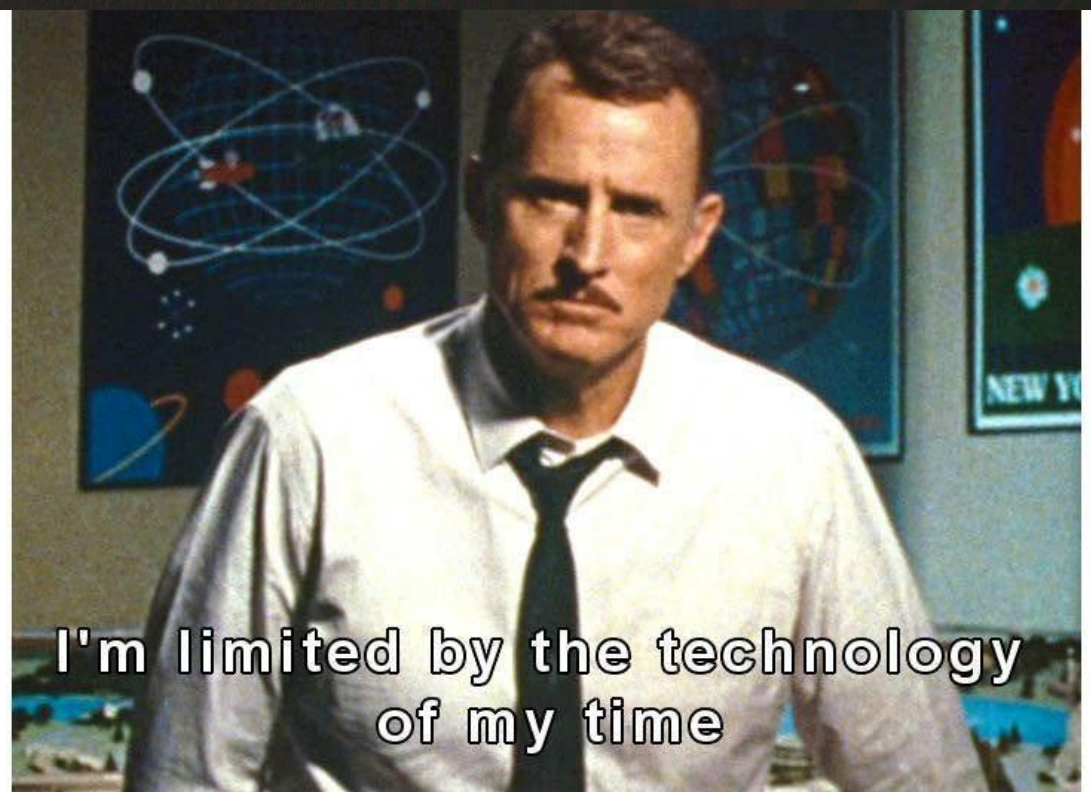
- r = 16 # Fine-tune 8, 16, 32, 64, 128
- lora_alpha = 16 # Fine-tune 8, 16, 32, 64, 128
- lora_dropout = 0 # If overfitting occcurs

Model: Pretrained encoder transformer

llama-3-8b-bnb-4bit



OutOfMemoryError: CUDA out of memory. Tried to allocate 1002.00 MiB. GPU 0 has a total capacty of 14.74 GiB of which 52.44 MiB is free. Process 3347037 has 4.65 GiB memory in use. Process 3320759 has 4.64 GiB memory in use. Process 3377253 has 5.40 GiB memory in use. Of the allocated memory 4.72 GiB is allocated by PyTorch, and 22.96 MiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting max_split_size_mb to avoid fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



Training: Out of Memory issues

- Running Training on GPUHub failed
  - Decreasing batch size to 1
  - Restarting session, emptying cache
  - Trying out different models and different approaches

- Running Training on Google Colab succeded
  - Limited time left
  - Limited access (I need to pay in order to use a GPU after some hours)

Config (Supervised Fine- tuning Trainer)

- gradient_accumulation_steps = 4
- max_steps = 500
  - num_train_epochs= 3
  - per_device_train_batch_size = 2
  - gradient_accumulation_steps = 4
- learning_rate = 2e-4, # Fine-tune 1e-5 to 5e-5
- lr_scheduler_type = "linear"
- warmup_steps = 5
- Loss Function: CrossEntropy
- Optimizer: AdamW_8bit with weight_decay: 0.1

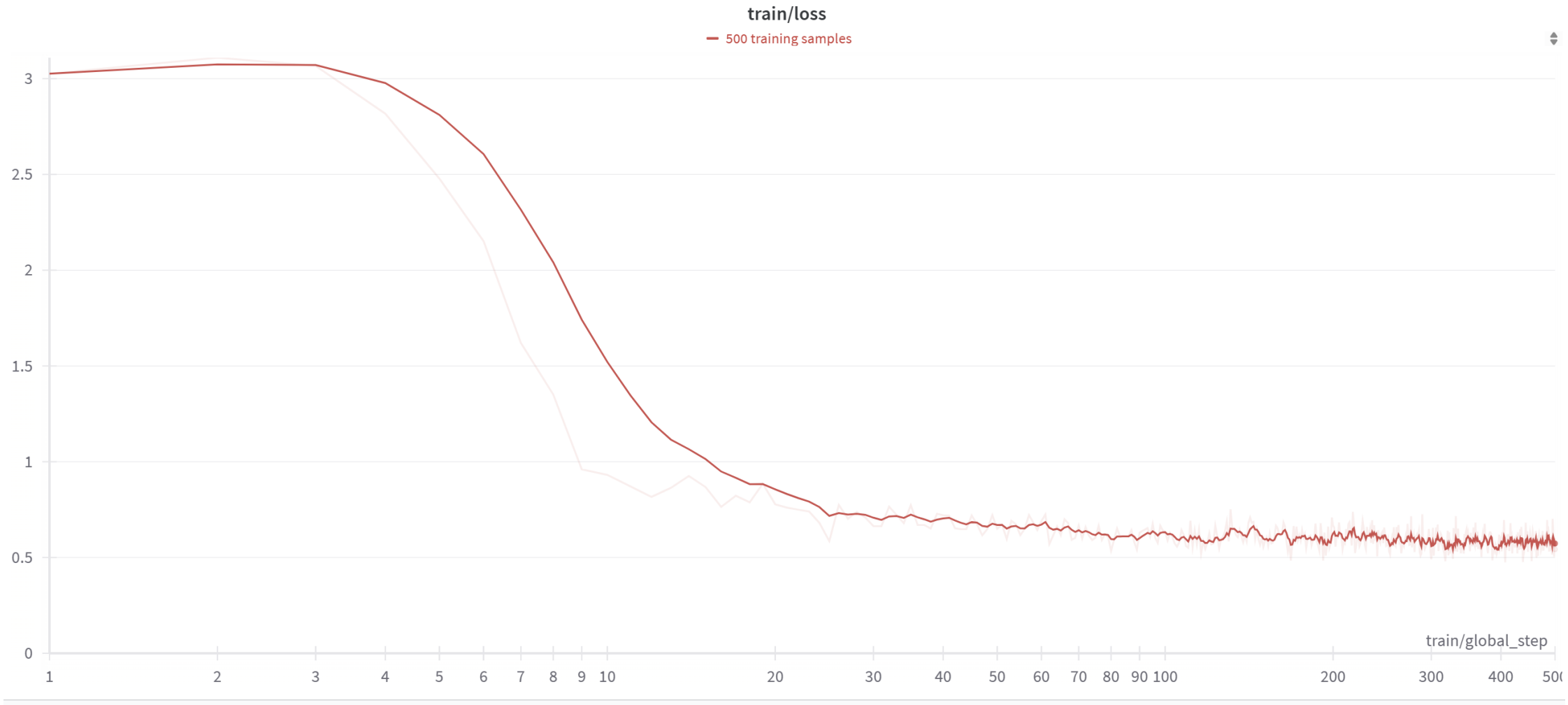
Results

«Best model»

- “Validation accuracy”: **14-48 %** (30-500 train and 50-100 validation samples)
  - project3 50.9%
  - Project4 55.3%

- “Test accuracy”: 53% (30 samples)
  - Project3 50.0%
  - Project4 66.2%

Train loss ☺


train/loss
500 training samples
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9
10
20
30
40
50
60
70
80
90
100
200
300
400
train/global_step

No valid Results

Problems

- No finetuning due to time constraints
- Only running small subsets
  - 1 epoch runs for 11 hours
- Results not representative with such small subsets

➔Only valuable insight which gives hope is the constantly decreasing loss

Conclusions

Interpretation

Potential success

- Looking at the loss it might work

Further Steps

- Longer training with all data
  - Maybe even bigger dataset (xl)
- More finetuning
  - learning rate, LoRA values)
- Prompt engineering for format
- Adjusting classification logic
  - Not only look at last token

Lessons Learned

Integrating PEFT

Handle Out of Memory issues

- Working with big models can quickly take up a lot of memory usage and it is hard to optimize
  - Try other (more powerful) gpus

Worse performance than base model

- Llama3 (0-shot) accuracy on winogrande task is 70.1%