Instructions for ACL-2015 Proceedings

Weiming Luo
Student id:mc464633
University of Macau
Department of data science
Macau, China
mc46463@um.edu.mo

Abstract

This modeling project used different languages to perform transfer learning on the English pre-trained model llama2 to allow the model to continue the story in different languages, with good results

1 Introduction

The purpose of this project is to use the pre-training model llama2 for story generation, where the model is given a prompt for the start of the story and the model follows the prompt to continue the story. The model has been pre-trained on the English dataset, and now the model is subjected to transfer learning so that it can continue the story in Chinese and Portuguese.

2 Dataset and training

All the data set are .jsonl format, per line of data set is like

`{text:"Once upon a time,...."}`

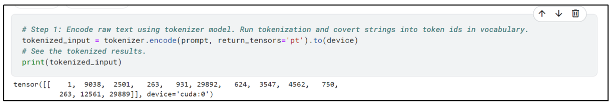
English, Chinese, Portuguese test data set 500 line, Chinese, Portuguese validation set 500 line, Chinese, Portuguese training set 10000 line. Additional Japanese dataset 400 for training, 10 for validation and testing. And training at free gpu server which support by Kaggle (Kaggle, 2023).

3 Methodology

3.1 Task 1 Decode dataset

The purpose of this section is to code the dataset, and the code for each word is the position of this word in the whole lexicon.Because this project uses pytorch as a framework, it returns the tensor in torch format.

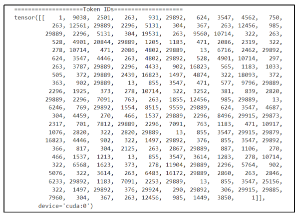
Model.generate() is the function which can continue the story, then generate the encode of the story



```
# Step 1: Encode raw text using tokenizer model. Run tokenization and covert strings into token ids in vocabulary.
tokenized_input = tokenizer.encode(prompt, return_tensors='pt').to(device)
# See the tokenized results.
print(tokenized_input)

tensor([[    1,  9038,  2501,   263,   931, 29892,   624,  3547,  4562,   750,
           263, 12561, 29889]], device='cuda:0')
```

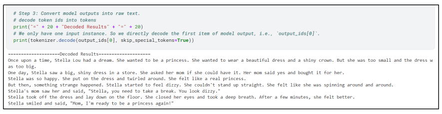
Figure 1: prompt encoding



```
====================Token IDs====================
tensor([[    1,  9038,  2501,   263,   931, 29892,   624,  3547,  4562,   750,
           263, 12561, 29889,  2296,  5131,   304,   367,   263, 12456,   985,
         29889,  2296,  5131,   304, 19531,   263,  9560, 10714,   322,   263,
           528,  4901, 20844, 29889,  1205,  1183,   471,  2086,  2319,   322,
           278, 10714,   471,  2086,  4802, 29889,    13,  6716,  2462, 29892,
           624,  3547,  4446,   263,  4802, 29892,   528,  4901, 10714,   297,
           263,  3787, 29889,  2296,  4433,   902, 16823,   565,  1183,  1033,
           505,   372, 29889,  2439, 16823,  1497,  4874,   322, 18093,   372,
           363,   902, 29889,    13,   855,  3547,   471,   577,  9796, 29889,
          2296,  1925,   373,   278, 10714,   322,  3252,   381,   839,  2820,
         29889,  2296,  7091,   763,   263,  1855, 12456,   985, 29889,    13,
          6246,   769, 29892,  1554,  8515,  9559, 29889,   624,  3547,  4687,
           304,  4459,   270,   466,  1537, 29889,  2296,  8496, 29915, 29873,
          2317,   701,  7812, 29889,  2296,  7091,   763,  1183,   471, 10917,
          1076,  2820,   322,  2820, 29889,    13,   855,  3547, 29915, 29879,
         16823,  4446,   902,   322,  1497, 29892,   376,   855,  3547, 29892,
           366,   817,   304,  2125,   263,  2867, 29889,   887,  1106,   270,
           466,  1537,  1213,    13,   855,  3547,  3614,  1283,   278, 10714,
           322,  6568,  1623,   373,   278, 11904, 29889,  2296,  5764,   902,
          5076,   322,  3614,   263,  6483, 16172, 29889,  2860,   263,  2846,
          6233, 29892,  1183,  7091,  2253, 29889,    13,   855,  3547, 25156,
           322,  1497, 29892,   376, 29924,   290, 29892,   306, 29915, 29885,
          7960,   304,   367,   263, 12456,   985,  1449,  3850,     1]],
       device='cuda:0')
```

Figure 2: encoding of answer

Then use decoder to decode the story form tensor to English. The principle is to use the index value of each word to find the corresponding word from the original thesaurus, and then convert it into English to print it out.



```
# Step 3: Convert model outputs into raw text.
# decode token ids into tokens
print("=" * 20 + "Decoded Results" + "=" * 20)
# We only have one input instance. So we directly decode the first item of model output, i.e., `output_ids[0]`.
print(tokenizer.decode(output_ids[0], skip_special_tokens=True))

====================Decoded Results====================
Once upon a time, Stella Lou had a dream. She wanted to be a princess. She wanted to wear a beautiful dress and a shiny crown. But she was too small and the dress w
as too big.
One day, Stella saw a big, shiny dress in a store. She asked her mom if she could have it. Her mom said yes and bought it for her.
Stella was so happy. She put on the dress and twirled around. She felt like a real princess.
But then, something strange happened. Stella started to feel dizzy. She couldn't stand up straight. She felt like she was spinning around and around.
Stella's mom saw her and said, "Stella, you need to take a break. You look dizzy."
Stella took off the dress and lay down on the floor. She closed her eyes and took a deep breath. After a few minutes, she felt better.
Stella smiled and said, "Mom, I'm ready to be a princess again!"
```

Figure 3: decode of answer

Then change some of the parameters, unlike the last generation, this time we chose to sample randomly and set the temperature to 0.6, indicating that the generated text has some randomness but is still generally skewed towards determinism (the higher the temperature the more random it is, the lower it is the more we tend to favor more probable answers), The top_p and top_k parameters were then adjusted as prompted, which together can determine the diversity and randomness of the sampling process, and when these two parameters were turned up to lower p to lower k, the model generated statements that could withstand more logical scrutiny. However, these few parameter changes didn't cause too many problems with the model's logic for generating the articles, but when I tried to change the prompt to a format and emotional coloring that wasn't quite the same as in the dataset (unlike the once upon a time there was a day with this kind of formatted beginning, and not the happy coloring of a fairy tale), the logic for generating the story started to get messed up. Overall the model does a good job

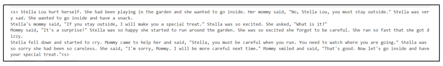


Figure 4: Enter Caption

with English stories, when trying other languages such as Noon, the model doesn't generate along with Chinese, but instead generates English stories that have nothing to do with prompts

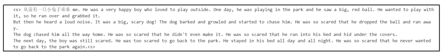


Figure 5: Sample of Chinese prompt

3.2 Task2 Perplexity

Firstly, the model evaluation functions Perplexity of this model is introduced, which in probabilistic terms is the average uncertainty of the language model with respect to the test data. t is the number of tokens in the sequence $P(x_i)$is the probability that the model will give the t token when it is generated. If perplexity is low, it means that the model predicts the data more accurately.

$$\text{Perplexity} = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log P(x_i \mid x_{<i})\right)$$

Dataset was test for four different languages and calculated their perplexity values, we found that except for English, which is very low, the perplexity values of Japanese, Chinese and Portuguese are very high, probably because the model is a pre-training model of English, which is a good fit for English grammar, and Portuguese, which is also made up of alphabets, has a higher degree of similarity to English.The perplexity values are also perplexity values are also lower than perplexity values are lower than those of Japanese.The perplexity values are lower than those of Japanese and Chinese.

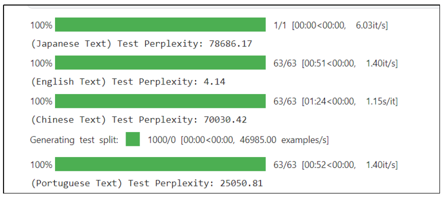


Figure 6: perplexity of four language

3.3 Task 3 Chinese transfer learning

The purpose of this task is to perform transfer learning on the model, and we will use this pretrained model for English to train Chinese and other languages. Firstly, we used default parameters to train Chinese data set for 8 epochs, At the end, the loss value has converged and looks good, so let's experiment with the model

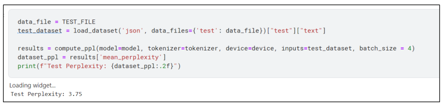


Figure 7: Perplexity of Chinese

After training the model, Chinese perplexity of the model is test and find that it is reduced to the same level as English, indicating that the model has a full understanding of Chinese. I test the model by giving it a Chinese prompt, and find that the model can already continue the story in Chinese.

Figure 8: Enter Caption

We gave a Chinese prompt to test the model, and found that the model can already be used in Chinese to continue the story, we changed the parameters of the generator such as task1, and found that the model's results gave the expected answers, but in general the logic is not as smooth as in English, probably because the difficulty of the Chinese language is greater than that of the English language.

For the control experiment, the learning rate of the model was tuned to 3e-5, and the learning rate was lowered to see how it affects the losses

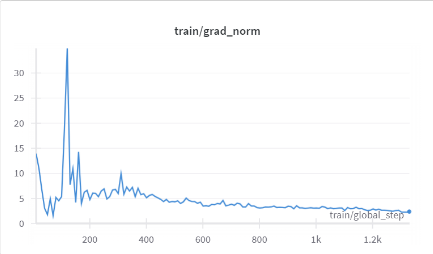


Figure 9: grad normalization of lr 1e-4

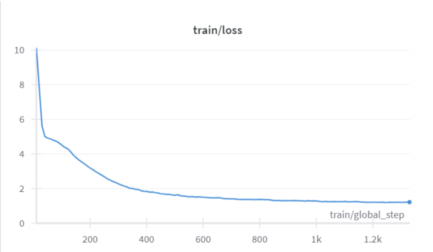


Figure 10: training loss of lr 1e-4

After 8 epochs of training, the model's $grad_norm$ is still in a fluctuating state and has not been reduced to 0, and the loss is larger than the first training, which proves that 8 epochs of learning after reducing the learning rate is not enough.In re-testing, the model gives answers that are also illogical.

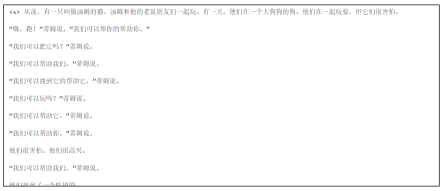


Figure 11: rave of Chinese model

3.4 Fonts

And model did not give the answer I want. Therefore there 8 epochs additional training for this model, when epochs attach 16, loss decay to 1.29, as same as learning rate 1e-4. And the Chinese perplexity value of the model has been reduced to 4, and the model has a good understanding of the Chinese language as well

3.5 Task 4 other language fine tune

Since there is no existing Japanese dataset, I asked ChatGPT to help me generate some data, 428 for the training set and 9 for both the test and validation sets, just for fun. Since the dataset is small, I

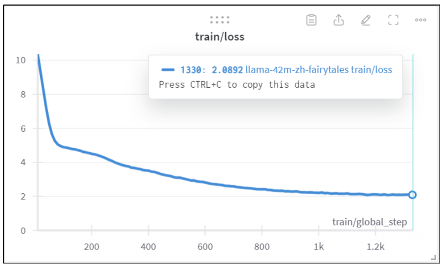


Figure 12: grad norm of lr 3e-5

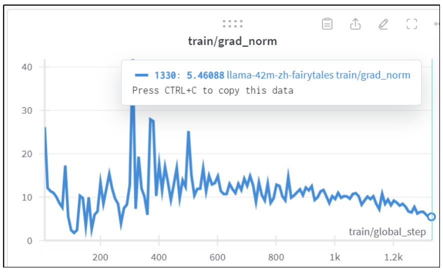


Figure 13: training loss of lr 1e-4

Figure 14: rave of Japanese model of 90 epochs

took a lower learning rate of 3e-5, and 90 epochs of training for such an approach.

Figure 15: Enter Caption

But for the result, it seems so stupid that always repeat one sentence. So the epochs changed to 300, the result seems better than before, it can generate a short story with logical sentence. What's more, the $grad_norm$ and loss value has dropped to nearly zero.

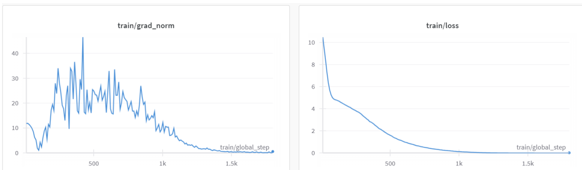


Figure 16: grad and training loss of JP datset

However, because of limit training set(only 400), Japanese model can not understand the

```
data_file = JP_TEST_FILE
test_dataset = load_dataset('json', data_files={'test': data_file})["test"]["text"]

results = compute_ppl(model=model, tokenizer=tokenizer, device=device, inputs=test_dataset, batch_size = 4)
dataset_ppl = results['mean_perplexity']
print(f"Test Perplexity: {dataset_ppl:.2f}")

Generating test split: 9/0 [00:00<00:00, 501.48 examples/s]
100% 3/3 [00:00<00:00, 13.92it/s]
Test Perplexity: 3.11
```

Figure 17: Test Perplexity of JP dataset

```
prompt = "昔々、ある日"

# Decoding hyperparameters
max_new_tokens = 300
do_sample = True
temperature = 0.3

tokenized_input = tokenizer.encode(prompt, return_tensors="pt").to(device)
output_ids = model.generate(
    tokenized_input,
    max_new_tokens=max_new_tokens,
    eos_token_id=1,
    do_sample=do_sample,
    temperature=temperature,
)
output_text = tokenizer.decode(output_ids[0])
print(output_text)

<s> 昔々、ある日、小さな村に勇敢な少年が住んでいました。彼の名前はタロウで、村の外にある深い森にいつも探検に出かけました。ある日、タロウは不思議な光を見つけ、その光を追いかけて行きました。すると、そこには大きな魔法の湖がありました。湖のほとりに座っていると、突然、湖の中から美しい妖精が現れました。<s>
```

Figure 18: normal JP prompt

prompt out of data set, I give a different prompt, but got a totally irrelevant answer. Therefore, to model a better understanding of a language requires a much larger collection of data thesaurus

```
prompt = "君の名は"

# Decoding hyperparameters
max_new_tokens = 300
do_sample = True
temperature = 0.3

tokenized_input = tokenizer.encode(prompt, return_tensors="pt").to(device)
output_ids = model.generate(
    tokenized_input,
    max_new_tokens=max_new_tokens,
    eos_token_id=1,
    do_sample=do_sample,
    temperature=temperature,
)
output_text = tokenizer.decode(output_ids[0])
print(output_text)

<s> 君の名は、ある日、遠い語り力と彼の名前はオリオンで、狐が住んでいました。彼の名前はシロで、森の森中の森の中を彼は旅をしていました。ある日、森に危機が訪れ、シロはその知恵を使って森を守ることを決意しました。彼は皆と協力し、無事に森を守りました。モモの知恵と勇気は、動物たちに感謝され、彼の名は森の守護者として語り継がれることになりました。<s>
```

Figure 19: different JP prompt

3.6 Use of Generative AI

This project used generative AI in the following, first ChatGPT helped to generate the Japanese dataset, gave him the appropriate text format and content requirements, and each time he asked it to generate ten 7-sentence long datasets, and then this was repeated 40 times. Secondly let GPT explain many parameters in the model such as top_k,temperature etc. Then also had it explain the use of the Perplexity function and how it is calculated.

Acknowledgments

First of all, I would like to thank Prof. Wong for giving me this opportunity to learn the training of llm model, so that I can understand how the pre-training and fine-tune works. Secondly, I would like to thank kaggle for providing a free training environment and 30 hours of GPU usage, which drastically reduced my training time!

References

Kaggle. 2023. Example Data on Kaggle. https://www.kaggle.com/dataset/example-data. Accessed: September 13, 2024.