Story-Based QA GPT Model

Anonymous Author(s)
Affiliation
Address
email

Abstract

NLP is of great significance, plenty of models are developed to achieve better performance in different tasks. Gpt-2 model is one of them with classic structure of the transformer, and reach great accomplishment.However, they generally acquire hundreds of millions of parameters. In order to explore whether language model can performe well with less paramerters. A language model with 3.6M parameters was built and trained. The preformance of pretrained model and finetuned model were analysized by gpt-3.5 in grammer and logic. The results shows that models with small parameters can only learn a part of grammar and part of the template to answer specific questions. This work sheds light on understanding of train procession and growth of language model. The language will learn grammer first and than basic structure to answer questions and finally logic if trained properly.

1 Introduction

Natural Language processing(NLP), is the field for allowing computers to understand human language. Besides decipher the meaning of every single word, the target for NLP is to comprehend the full context of human language.

There are planty of NLP tasks, including:

- Classifying whole sentences
- Classifying each word in a sentence
- Generating text content
- Extracting an answer from a text
- Generating a new sentence from an input text
- ...

In my project, I build a text generation model using Huggingface API.[1] . The architecture is the classic gpt-2 model. In order to decrease the number of parameters to train faster, the layer width and depth are downscaled. Tiny Story is used to pretrain the model, while alpace dataset is applied to do the instruction tuning. Evaluation process is conducted by perplexity. For further evaluation, the gpt-3.5 is applied according to the method provided in Eldan and Li [2023] to evaluate the story telling level of the pretrain model. In order to evaluate the finetuned model, the comment by gpt-3.5 and specific analysis on grammer and logic is conducted. Generally, the language model with small number of parameters can only achieve partial correction on grammar and can hardly showcase any logic on the task, only relationships between words are learned.

[1] `https://huggingface.co/`

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

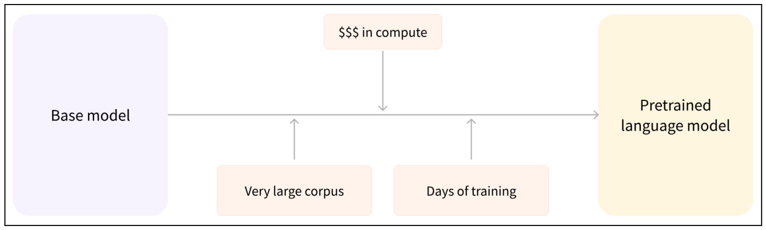

Figure 1: The pretraining pipeline

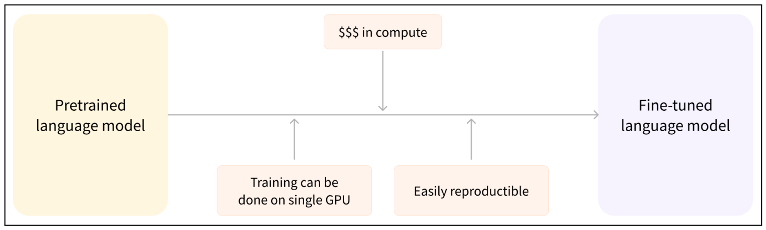

Figure 2: The finetune pipeline

2 Transfer Learning

The idea of Transfer Learning is to leverage the knowledge acquired by a model trained with lots of data on a another task. Assuming the model A has been specifically trained for task A with huge dataset and I right now have a task B to tackle. One option is to train the model from scratch, which would possibily consume plenty of energy and cause carbon emission. However we could initializing the model B using the weights from model A, which is a time-and-energy-saving way to "Transfer" a existing model to a new one. Apart from time and cost concern, using pretrained model usually have a better performance as the previous model has been trained on large scale of corpus which helps it to gain a fundamental understanding of the field. The figures shows the process of transfer learning.[2]

3 Model architecture

The basic architecture is same to GPT-2 Model. From my perspective the GPT-2 model is a combination of the idea of RNN and attention mechanisms. Transformer is the main model architecture. According to Vaswani et al. [2017], the transformer allows for significantly more parallelization and can reach a new state of the art in performance in translation.

3.1 Basic Recurrnet Neural Networks

In order to deal with sequential input, the net works are required to possess the current input while considering the former ones. This is how RNN works.See figure 3.[3]

If we do not consider the cycle with hidden parameter v, the net work is just a fully connected neural networks. However, when the sequence itself is important, we can expand the structure in a new demension, by a corresponding time sequence.

The basic iterate functions are as follows

$$O_t = g(w \cdot h_t)$$

$$h_t = f(U \cdot X_t + W \cdot S_{t-1} + b_t)$$

[2] https://huggingface.co/learn/nlp-course/chapter1/4?fw=pt

[3] https://en.wikipedia.org/wiki/Recurrent_neural_network

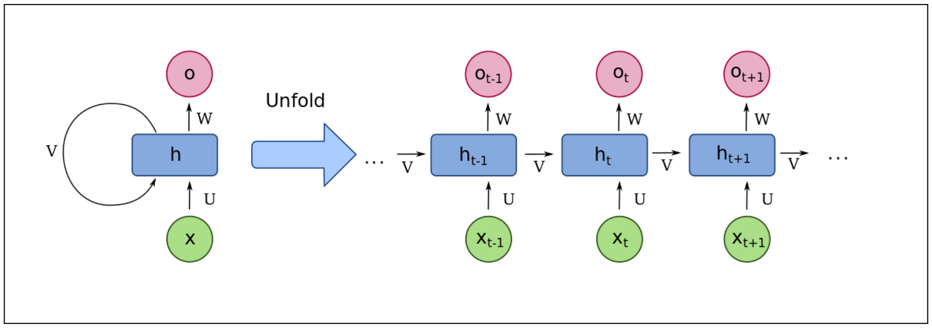

Figure 3: The basic architecture of RNN

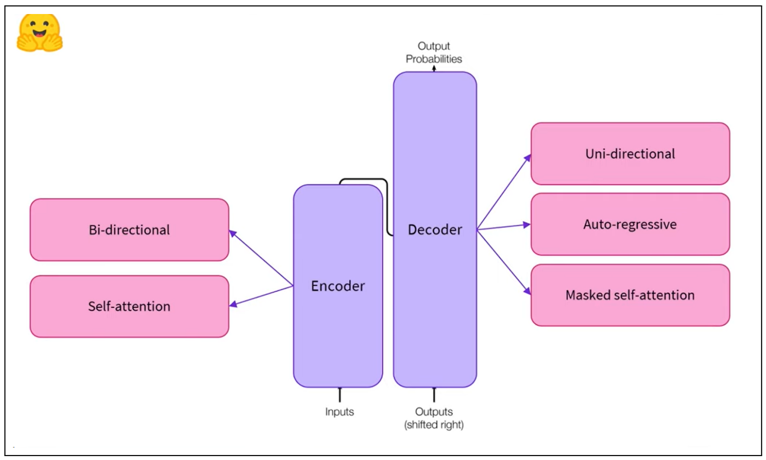

Figure 4: The basic structure of transformer architecture

3.2 Transformers

3.2.1 General structure

The figure 4 shows the basic structure of a transformer. Encoder deals with the input data via bi-directional way, in which the full data can be processed, or be seen by the encoder. While the decoder can only see the previous data, which is a uni-directional masked self-attention mechanism. Similar to RNN, this is also a time-sequenced structure. Howeverm, the output from the previous decoder may become the input to the next decoder.

3.2.2 Encoder

As shown in figure 6 and 7,the encoder take word vectors(a numerical representation for words, maybe a one-hot vector) in and output the feature vectors, or overall a feature tensor. The feature vector contain the information of the position of the word. The overall feature tensor contains the information of the whole context.

However, the algorithm can only process numbers or vectors, while the input is acutally words. Consequently, the texts need to experience a series of **data preprocessing** or **embedding**.

Tokenization

A sentence is composed of a sequence of words, so the first thing is to split the sentence into a series of **tokens**, the single input. There are several ways of tokenization.

- Word-based tokenization: Convert a sentence to words. This is straight forward but not good enough as the words sometimes are composed of subwords. For instance, "Transformer" is composed of "Transform" and "-er".

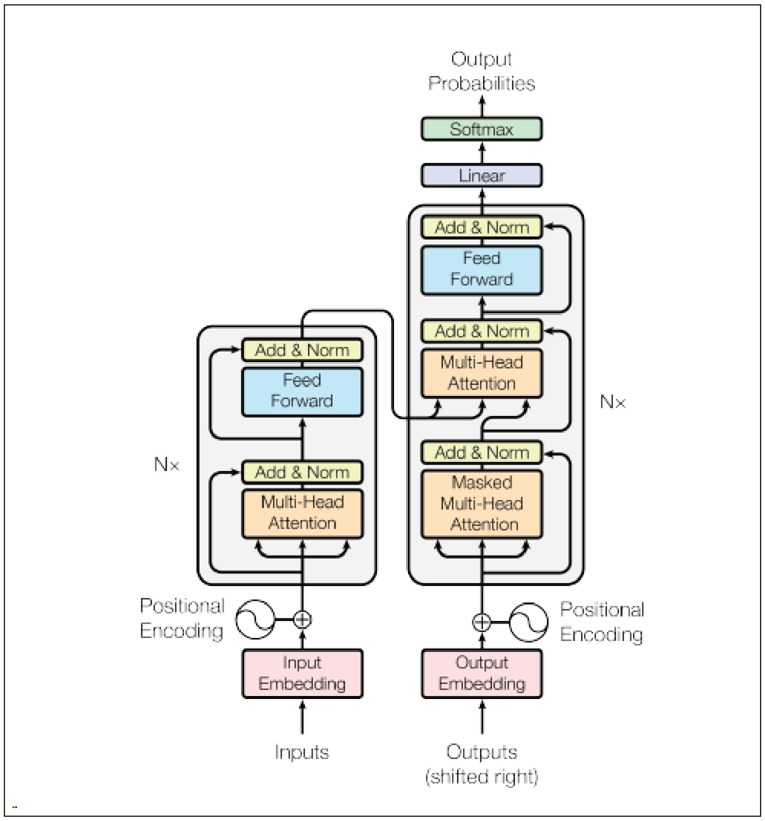

Figure 5: The detail of a transormer

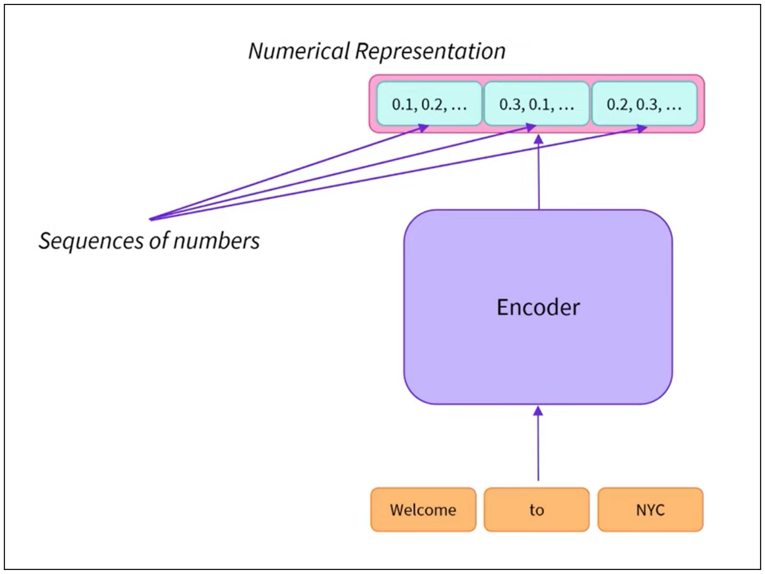

Figure 6: The input & output of a encoder

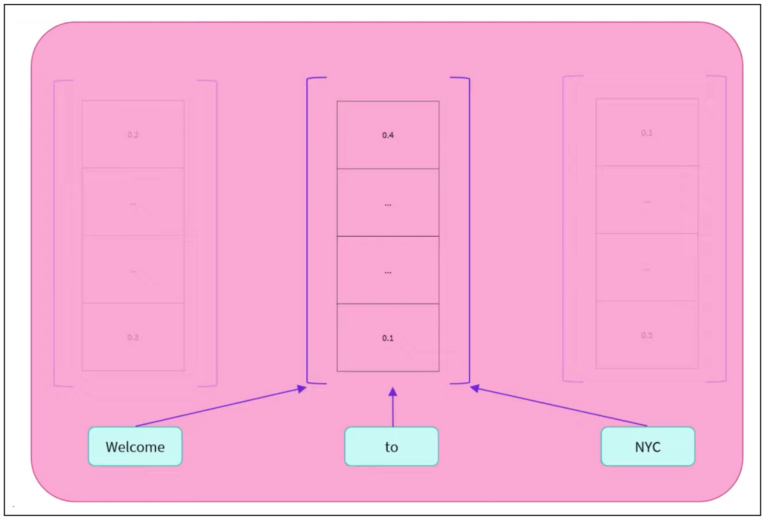

Figure 7: A detail look to feature vectors

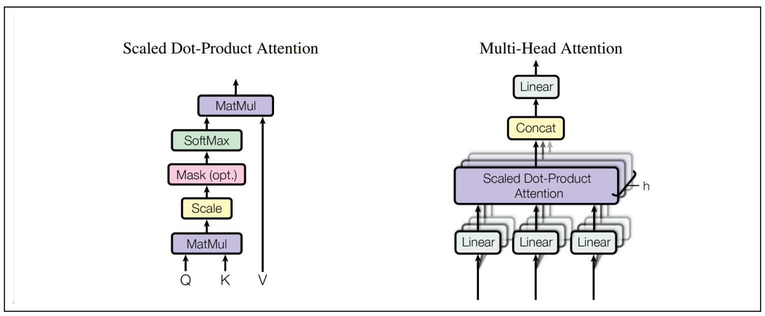

Figure 8: The attention mechanism and multi-head attention

- Character-based tokenization: Convert a sentence to every character, this is specailly useful for language like Chinese.
- Subword-based: this is mostly used version for Latin linguistic group.

After tokenization, it is time to convert the input into numbers. This is done by converting every subword to a number according to the vacubulary. There are several algorithm to build vocabulary and conduct the convert.[4]

- Byte-pair Encoding tokenization.
- WordPiece tokenization.
- Unigram tokenization.

3.2.3 Self-Attention

Now comes to the most important part in a transformer. The self-attention mechanism. The basic idea of attention mechanism is to calculate the relationship between different words, the **attenion** of each other.

But how?

Firstly it is resonable to give a value to each word, which is essential to calculate. In order to access a value the "**Query**" is needed. Correspondingly, in order to be accessed the "**Key**" is needed. As a result, the programme will automatically generate a query, key, value pair for each token(the input vector from input_id a).

[4] `https://huggingface.co/learn/nlp-course/chapter6/4?fw=pt`

This can be done by :

$$q_i = W_q a_i$$
$$k_i = W_k a_i$$
$$v_i = W_v a_i$$

The function W can be learned which actually increase the representativeness of the network.

Secondly, it is time to calculate similarity, which is conducted by Scaled Dot-Product Attention: For the first word, the attention to the ith word is:

$$\alpha_{1,i} = q_1 \cdot k_i / \sqrt{d}$$

where d is the demension of d and k, which scales the representation, as the more dimension the larger the number.

The attention actually is a probability to have a specific word in a given context, so it is essential to convert output to probability. Then the softmax function is applied.

$$\hat{\alpha}_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$

As a result, for the first word, this could generate a vector $\hat{\alpha}_1$(after softmax), for the whole sequence of tokens, this could generate a matrix, attention matrix $\hat{A}$.

$$\hat{A} \ll A = \vec{q} \cdot \vec{k}^T$$
$$\begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \dots & \hat{a}_{1,n} \\ \hat{a}_{2,1} & \hat{a}_{2,2} & \dots & \hat{a}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{a}_{n,1} & \hat{a}_{n,2} & \dots & \hat{a}_{n,n} \end{bmatrix} \ll \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \cdot \begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix}$$

To get the whole information, the values need to multiply the probability which is the prediction.

$$b_1 = \sum_i \hat{\alpha}_{1,i} v_i$$

This can also be presented to vector form

$$\vec{b} = \hat{A} \cdot \vec{v}$$
$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \dots & \hat{a}_{1,n} \\ \hat{a}_{2,1} & \hat{a}_{2,2} & \dots & \hat{a}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{a}_{n,1} & \hat{a}_{n,2} & \dots & \hat{a}_{n,n} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

This is what is demonstrated in left part of figure 8 from Vaswani et al. [2017].

However, during the calculation the position information is eliminated through multiplication. As a result, the information needs to add manually.

The figure 5 (Vaswani et al. [2017]) shows the pipline after embedding. The position encoding add the position information through e_i.

Originally,

$$x_i \gg a_i \gg \begin{bmatrix} q_i \\ k_i \\ v_i \end{bmatrix}$$

After position encoding:

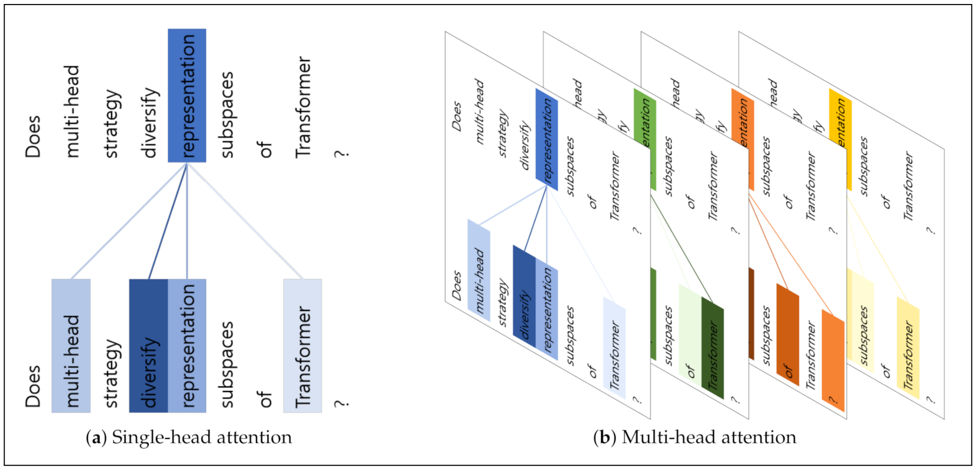

Figure 9: The comparison of single-head attention and multi-head attention

$$x_i' \gg a_i + e_i \gg \begin{bmatrix} q_i' \\ k_i' \\ v_i' \end{bmatrix}$$

This is done by enlongation of x_i by an one-hot position vector

$$p = [0, \quad 0, \quad \dots, \quad 1, \quad 0, \quad \dots, \quad 0,]^T$$

Right now the input vector becomes

$$x_i' = \begin{bmatrix} x_i \\ p_i \end{bmatrix}$$

After a linear transfromation

$$a_i + e_i = M \cdot x_i' = [M_x, M_p] \cdot \begin{bmatrix} x_i \\ p_i \end{bmatrix}$$

Although the matrix M can be learned, the results is not good.

In this self-attention mechanism, the attention matrix is symmetrical, which means when calculating the results vector $\vec{b}$, the information from full sentence is considered.

Masked self-attention

However, this is not true for masked attention. It is reasonable to consider only the information before when generating the next words. That is to say when we predict the next word, we only have the attentions between known words. For example, when we predict the fourth word, only $\alpha_{1,1}, \alpha_{2,(1,2)}, \alpha 3, (1, 2, 3)$ exist. There should be nothing exist in the upper triangular part of $\hat{A}$. To achieve this, we can set the upper triangular part to $-\infty$ as after softmax function the value should be zero.

Multi-head self-attention

One [q,k,v] set can only learn one kind of attention. As figure 8 and 9(Yun et al. [2021]) demonstrated however there maybe plenty of links between different words including semantic relationship and syntax rules. Consequently, multihead attention is essential to deal with the problem.

3.2.4 Decoder

A decoder bascially has similar components as an encoder, but with a musked attention mechanism.

4 Pretrain Process

The reported pretrained model is available on huggingface hub. [5]

Other unreported pretrained model is listed in Appendix

4.1 Dataset:Tinystories

Pretrain dataset was Tinystories [6]. It is a synthetic dataset of short stories that only contain words that a typical 3 to 4-year-olds usually understand, generated by GPT-3.5 and GPT-4. TinyStories can be used to train and evaluate LMs that are much smaller than the state-of-the-art models (below 10 million total parameters), or have much simpler architectures (with only one transformer block), yet still produce fluent and consistent stories with several paragraphs that are diverse and have almost perfect grammar, and demonstrate reasoning capabilities,(Eldan and Li [2023]).

4.2 Model Settings

The model has the same archtecture as the gpt-2 model, with some differences in parameters. The total parameter number of the model is 3.6M.

- Model Architecture Parameters
  - architectures: ['GPT2LMHeadModel']
  - model_type: gpt2
  - vocab_size: 50257
  - **n_embd: 64 (Hidden size of the layers)**
  - **activation_function: gelu_new**
  - **n_layer: 6 (The number of hidden layer)**
  - **n_head: 8 (The number of attention heads)**
  - resid_pdrop: 0.1 (The dropout probability)
  - embd_pdrop: 0.1
- Sequence Parameters
  - n_positions: 1024 (Max position embeddings: The maximum sequence length the model can handel)
  - layer_norm_epsilon: 1e-05
  - initializer_range: 0.02
- Attention Mechanism Parameters
  - attn_pdrop: 0.1
  - scale_attn_weights: True
  - scale_attn_by_inverse_layer_idx: False
  - **n_ctx: 128 (Context length)**
- Token parameters
  - bos_token_id: 50256
  - eos_token_id: 50256

4.3 Training details

Pretrain process was conducted using Huggingface API, and the training process was monitored by Weight and Bias[7]-Huggingface hub API

The training parameters are demonstrated below:

- per_device_train_batch_size=32,

[5] `https://huggingface.co/Toflamus/GPT-2_para3M`

[6] `https://huggingface.co/datasets/roneneldan/TinyStories`

[7] `https://wandb.ai/home`

- per_device_eval_batch_size=32,
- evaluation_strategy="steps",
- eval_steps=100,
- logging_steps=100,
- gradient_accumulation_steps=8,
- num_train_epochs=1,
- weight_decay=0.1,
- warmup_steps=100,
- lr_scheduler_type="cosine",
- learning_rate=5e-4,
- save_steps=100,
- fp16=True,
- report_to="wandb",

The learning rate varied with train process, it is demonstrated in figure 10:

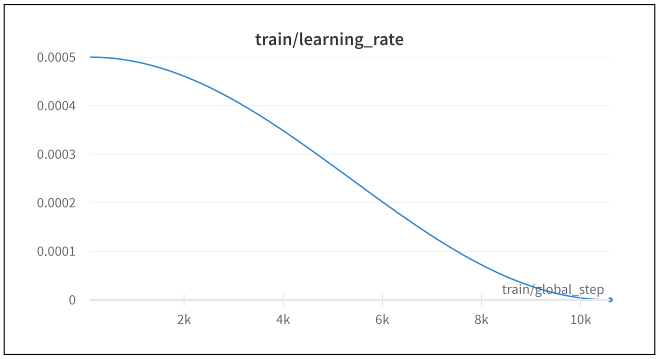

Figure 10: Pretrain learning rate

The train loss and validation loss is reported in figure 11 and 12:

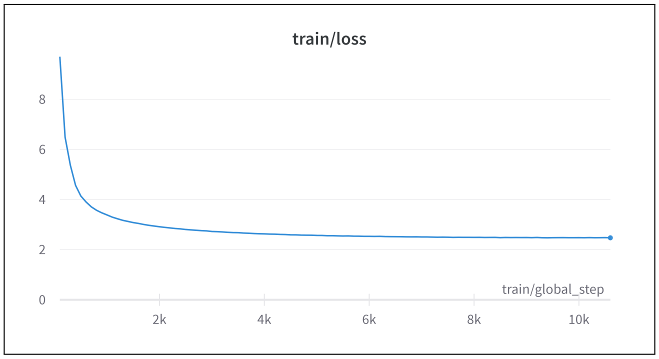

Figure 11: The loss curve on train datasets

4.4 Evaluation of Pretrained Model

Perplexity As the relationship of perplexity and evaluation loss is demonstrate in Appendix, the figure for perplexity is basicly the same with the evaluation loss on test dataset.

The final perplexity is around 10.1827.

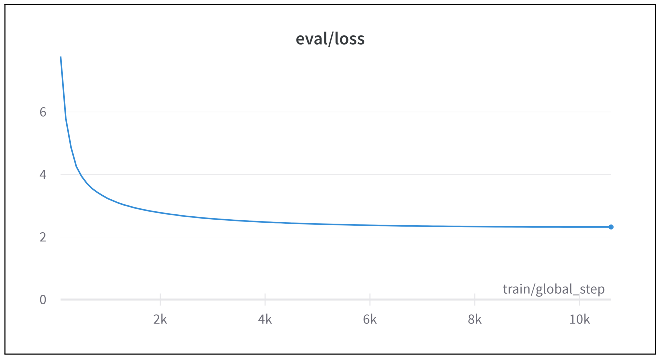

Figure 12: The loss curve validation datasets

Evaluation by GPT Reported in Eldan and Li [2023], the LLM GPT-4 can be used to evaluate some aspects of language model.

Due to finacial factors, GPT-3.5 is applied to evaluate the language model. GPT-3.5 was prompted to first provide a verbal assessment and then grade the completion. The transcript of interaction is as follows, with the parts produced by GPT-3.5 in italic font: The beginning of the story is from hugging face API "Examples". Generation process will not be finished until the website warns 'No text was generated' Following is an example:

> **The following exercise, the student is given a beginning of a story. The student needs to complete it into a full story.The exercise tests the studentt's language abilities and creativity. The symbol *** marks the separator between theprescribed beginning and the students completion:**
>
> My name is Clara and I am grandma." She thinks.
>
> "Sorry, grandma. We should have something else. I am my birthday. I am a queen," Mom says.
>
> Anna is angry and sad. She knows the story is not for her sister. She wants to share her poems in the room. It can write with the things. She says she will make Anna happy and Ben a card. He gives them some love books and a cat.
>
> **Please provide your general assessment about the part written by the student (the one after the *** symbol).Is it gramatically correct? Is it consistent with the beginning of the story? Pay special attention to whether thestudent manages to complete the sentence which is split in the middle by the separator ***.**
>
> *The part written by the student after the *** symbol is not grammatically correct, and it lacks coherence and consistency with the beginning of the story. There are several issues with the student's completion:*
>
> *1. Lack of Clarity: The student's completion lacks clarity and coherence. It is unclear what the student is trying to convey, and it doesn't follow a logical progression from the beginning of the story.*
>
> *2. Incomplete Sentence: The student's completion does not complete the sentence that was split in the middle by the separator ***. This creates confusion and disrupts the flow of the narrative.*
>
> *3. Lack of Connection: The student introduces new characters (Anna, Ben), a mention of poems and a cat, which are not mentioned or established in the beginning of the story. This creates inconsistency and confusion.*
>
> *4. Grammatical Errors: There are grammatical errors throughout the completion, making it difficult to understand. For example, "It can write with the things" is not a grammatically correct sentence.*
>
> *In summary, the student's completion is incoherent, lacks clarity, and does not follow the beginning of the story. It would benefit from a complete rewrite to align with the initial narrative and improve its grammatical correctness and clarity.*
>
> **Now, grade the students completion out of a maximum of 10 points in terms of grammar, creativity, consistency with the storys beginning and whether the plot makes sense. Moreover, please provide your best guess of what the age of the student might be, as reflected from the completion. Choose from possible age groups: A: 3 or under. B: 4-5. C: 6-7. D: 8-9. E: 10-12. F: 13-16.**
>
> *I would rate the student's completion as follows: Grammar: 2/10 - The completion contains several grammatical errors and incomplete sentences, making it challenging to understand.*
>
> *Creativity: 2/10 - While the student introduces new elements like poems, love books, and a cat, these additions do not fit well with the beginning of the story and do not contribute to a coherent narrative.*
>
> *Consistency with the story's beginning: 1/10 - The completion is highly inconsistent with the beginning of the story. It introduces new characters and concepts that were not mentioned initially.*
>
> *Plot coherence: 1/10 - The plot in the completion does not make sense and lacks a clear storyline. It does not build upon the beginning of the story or provide a meaningful continuation.*
>
> *Based on the completion and the writing style, it's challenging to pinpoint the exact age of the student, but it appears to reflect a very young writer, possibly in the age group A or B (3 or under or 4-5 years old). The language and narrative structure suggest a lack of developed writing skills and an understanding of story coherence.*

4 examples were chosen to define the proformance of the model. Comparison with gpt2(124M) was made on task of story generation. The results are demonstrated in figure 13. Conclusion can be made that gpt2 model gets better performance in grammer and the generated texts are more mature. However, in other aspects the model do not possess much edge over my model.

Additionally, gpt2 model usually generate repeatitive answer, however this will not happen in my model, as the example shown below:

> Prompt: Once upon a time, **gpt2**: Once upon a time, ***the world was a place of great beauty and great danger. The world was a place of great danger, and the

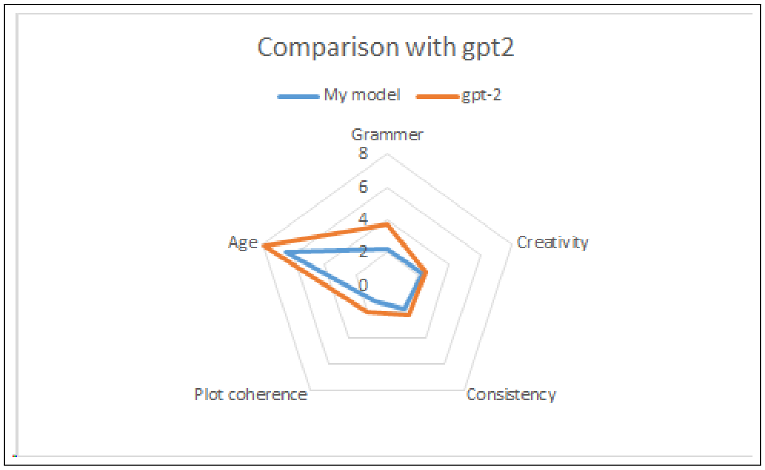

Figure 13

> world was a place of great danger. The world was a place of great danger, and the world was a place of great danger. The world was a place of great danger, and the world was a place of great danger.
>
> **my model** Once upon a time, there was a little girl named Lily who loved to draw with her cat. One day, she was playing outside and suddenly noticed something shiny in the garden. She wanted to build something in the distance. Kitty ran to her friend Tim and said, "Welcome. You have to make me happy." Lily said, "Yes, Tim. Don't be strong," "Ok I will fill with the box."

5 Finetune Process

5.1 Instruncion tuning dataset:Stanford Alpaca

The dataset and code for instruction tuning is from Taori et al. [2023] 's Git-hub website [8].

This produced an instruction-following dataset with 52K examples obtained at a much lower cost (less than $500). In a preliminary study, they also find our 52K generated data to be much more diverse than the data released by self-instruct. We plot the below figure (in the style of Figure 14 in the self-instruct paper to demonstrate the diversity of our data. The inner circle of the plot represents the root verb of the instructions, and the outer circle represents the direct objects.

The data was splited by 1/10 for train set and validation set. Besides, the data is in the form of text and they were put in the model throught the prompt templates:

Listing 1: Prompt template

```
PROMPT_DICT = {
"prompt_input": (
"Below is an instruction that describes a task, paired with an
input that provides further context. "
"Write a response that appropriately completes the request.\n\n"
"### Instruction:\n{instruction}\n\n### Input:\n{input}\n\
n### Response:"
),
"prompt_no_input": (
"Below is an instruction that describes a task. "
"Write a response that appropriately completes the request.\n\n"
"### Instruction:\n{instruction}\n\n### Response:"
),}
```

[8] https://github.com/tatsu-lab/stanford_alpaca

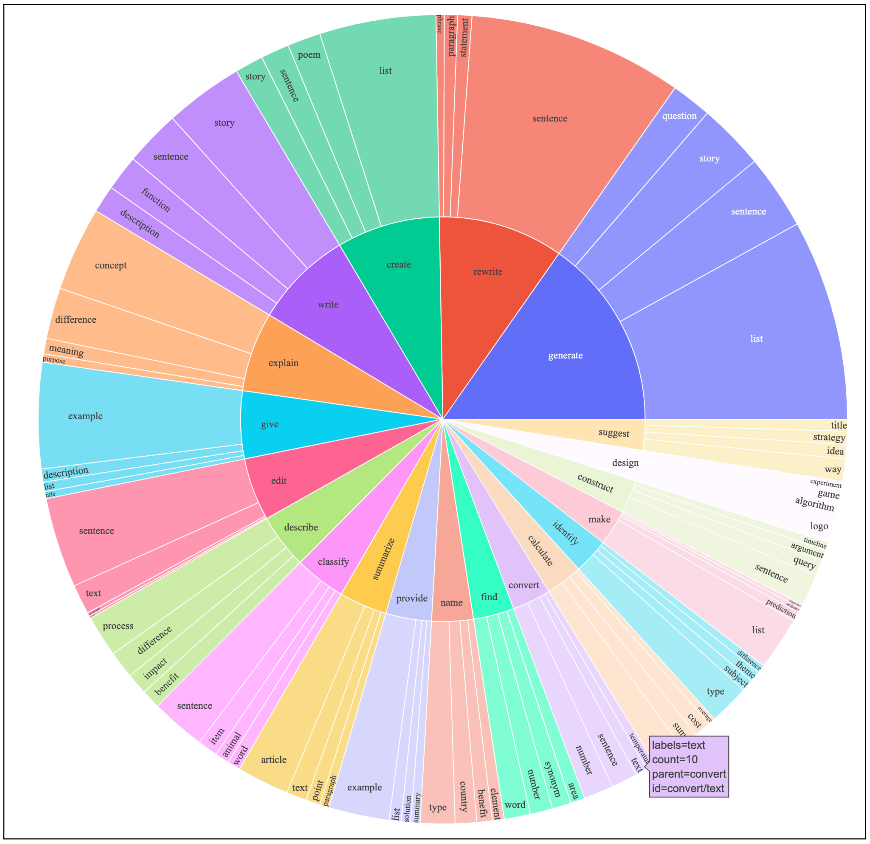

Figure 14: Parse analysis of the dataset

5.2 Finetune details

The **Tokenizer** is still the same tokenizer, and the training arguments are listed below:

- model_max_length: int = 512
- bf16: bool = True
- num_train_epochs: int = 3
- gradient_accumulation_steps: int = 8
- evaluation_strategy: str = "Steps"
- learning_rate: float = 2e-5
- weight_decay: float = 0.0
- warmup_ratio: float = 0.03
- lr_scheduler_type: str = "cosine"
- logging_steps: int = 1
- tf32: bool = True
- full_determinism = False
- seed = 42

The learning rate, train loss and eval loss are demontrated in figure 15,16 and 17

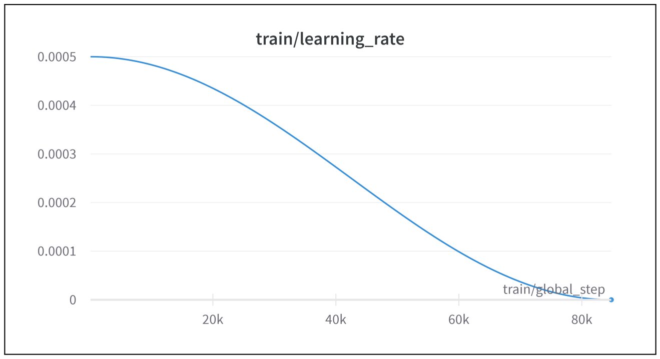

Figure 15: The fine-tuning learning rate

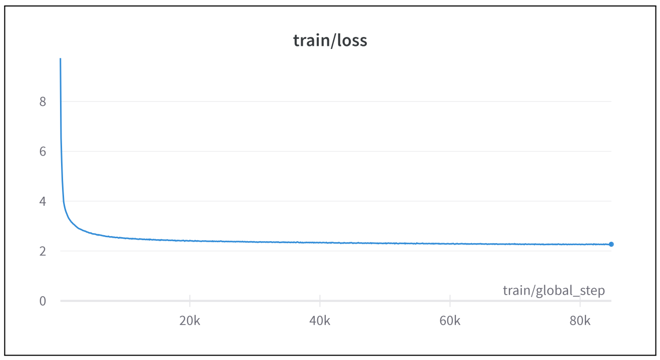

Figure 16: The loss curve on train datasets

5.3 Finetuned model evaluation

As the instruction tuning and pretrain used different dataset, we can imagine that the performence will not be desirable. Possibly, the stroy generation ablity will be largely kept while the ability to follow the instruction can hardly achieve.

Perplexity The PPL of the finetuned model on test dataset is 394.8 which is away far from ideal.

Text evaluation Now let's take a look at the performence on question answering ability with and without context.

QA without context The following section demonstrated the queation answering without context.

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Instruction: "Give three tips for staying healthy." Response: **And your task is always much to be caused for energy, than the size of any time. It is to have a strong choice of it that can be important in your work. In time, it's time that only can include a new activity and a safe for success."**

The bold text was the answer for the question. From my perspective, it is too hard for a 3.6M model to generate reasonable dialogue in a specific question,especially considering the fact that the model was even not trained with relative materials.

However, there can still be some sense for its answer. Words 'Energy','Activity','Safe' can be relevant to well-being. This means the model learned some words that are often used with 'healthy'. Besides, words 'choice','important' means the model was trying to use language in the context of giving suggestions. This means the model partly learned the appropriate way to give suggestions.

Objectively, GPT-3.5 gave a review to the output of my model

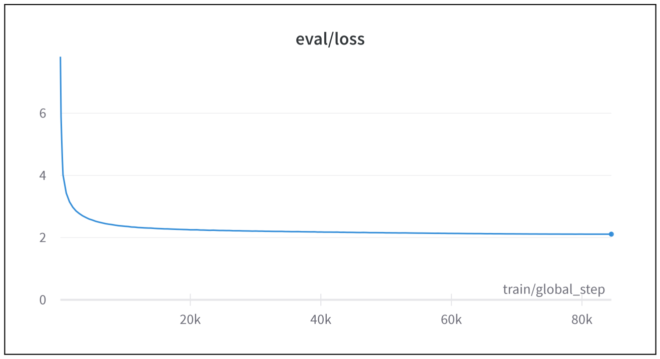

Figure 17: The loss curve on eval datasets

> The response provided in the context of the given instruction is not appropriate. It does not provide three tips for staying healthy but instead offers a vague and unrelated statement. The response does not address the task correctly.

QA with context The following section demonstrated the queation answering with context.

> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Instruction: Edit the following sentence to make it more concise. Input:He ran to the bus stop in order to catch the bus that was due to arrive in five minutes. Response: **The car was much better, including the bus had them. The bus is the first time to help the bus of the bus become extra creative and strong.**

For a QA instruction with context, the given answer is more specific, which needs logical thinking and context understanding.

The given answer:

> He ran to the bus stop, due to arrive in five minutes.

This was acutually too hard for my model, as it only learned some simple connections between words. The response from my language model was totally irrelevant to the instruction.

However, the we can know that the model did learned some links, as the response had words like ’car’,’bus’,which is relevant to bus stop and ’first time’, which is relevant to ’five minutes’.

For gpt2 the answer for the first instruction is:

> Give three tips for staying healthy.
>
> **1. Don’t be afraid to ask questions.**

It seems that the gpt2 model only learned the way to give advice but not with any logic. However, as there are more parameters in gpt2 model, and the training data is more diverse it can learn a better template on giving advice. Besides, there are less grammatical errors.

For gpt2 the answer for the second question is :

> Edit the following sentence to make it more concise. He ran to the bus stop in order to catch the bus that was due to arrive in five minutes.
>
> **He was in the middle of the bus when he saw a man with a gun.**

It seems that gpt2 can response to questions better, with fewer grammatical errors. However, it still does not learn any logic. However, it is one level higher than my model as it grasps correct grammer and the template to different questions.

6 Conclusion and Discussion

From the experiment above, language model with few paramaters can hardly learn much about language. There would be a lot of incoherence and grammar mistakes. However, with proper training dataset, the model can have comparable performance to large models. The meaning of small language model is that it can show the learning process of the model to us. For example, the model will learn basic grammar in the first place. If there are more parameters, the model will enhance its grammar and, when trained properly, learn the template to tackle different tasks. But still, the model won't be able to learn logic and coherence. This can only be achieved by real large language model.

7 Appendix

7.1 The relationship between Cross Entropy Loss and Perplexity

Perplexity [9] is defined as

$$PPL(p) := 2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)} = \prod_x p(x)^{-p(x)}$$

where $H(p)$ is the entropy (in bits) of the distribution, and x ranges over the events. The base of the logarithm need not be 2: The perplexity is independent of the base, provided that the entropy and the exponentiation use the same base.

In the special case where p models a fair k-sided die (a uniform distribution over k discrete events), its perplexity is k. A random variable with perplexity k has the same uncertainty as a fair k-sided die. One is said to be "k-ways perplexed" about the value of the random variable. Unless it is a fair k-sided die, more than k values may be possible, but the overall uncertainty is not greater because some values may have a probability greater than 1/k.

$$PPL(1/k) := 2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)} = \prod_{i=i}^{k} \frac{1}{k}^{-1/k} = k$$

For a perplexity modelof an unknown probability distribution p, may be proposed based on a training sample that was drawn from p. Given a proposed probability model$q$, one may evaluate q by asking how well it predicts a separate test sample $x1, x2, ..., xN$ also drawn from p. The perplexity of the model q is defined as

$$b_{-\frac{1}{N}\sum_{i=i^N} log_b q(x_i)} = (\prod_i q(x_i))^{-1/N}$$

where b is customarily 2. Better models q of the unknown distribution p will tend to assign higher probabilities q(xi) to the test events. Thus, they have lower perplexity: they are less surprised by the test sample.

Cross entropy:[10] The cross-entropy of the distribution q(Predicted) relative to a distribution p(Real) over a given set is defined as follows:

$$H(p, q) = -E_p[logq]$$

As the real distribution can hardly get,so the estimation is H(T,q) = -$\sum_{i=1}^{N} \frac{1}{N} log_2 q(x_i)$

As discussed above,

$$log(PPL) = H(T, q)$$

7.2 Other language models

They are avaiable on my home page[11]. Generally, there are not much changes. The lr schedular and max context length is changed for other pretrain models. While the number of epochs are changed for other finetuned models.

[9] https://en.wikipedia.org/wiki/Perplexity

[10] https://en.wikipedia.org/wiki/Cross-entropy

[11] https://huggingface.co/Toflamus

8 Acknowledgement

Thanks for teacher and TA for their concern,knowledge and understanding.

Thanks for my friend in Wuhan University for configuration of the server environment.

References

R. Eldan and Y. Li. Tinystories: How small can language models be and still speak coherent english?, 2023.

R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

H. Yun, T. Kang, and K. Jung. Analyzing and controlling inter-head diversity in multi-head attention. *Applied Sciences*, 11(4), 2021. ISSN 2076-3417. doi: 10.3390/app11041548. URL https://www.mdpi.com/2076-3417/11/4/1548.