AI51701/CSE71001 Assignment 3: Transformers and Pretraining

---

1 Pretrained Transformer models and knowledge access (70 points)

1.1 (a, 0 points) Check out the demo.

No answers for this part

1.2 (b, 0 points) Read through `NameDataset` in `src/dataset.py`, our dataset for reading name-birthplace pairs.

No answers for this part

1.3 (c, 0 points) Implement finetuning (without pretraining).

No answers for this part

1.4 (d, 10 points) Make predictions (without pretraining).

Model's accuracy on the dev set:

```
data has 418352 characters, 256 unique.
Correct: 5.0 out of 500.0: 1.0%
data has 418352 characters, 256 unique.
number of parameters: 3323392
437it [00:09, 45.16it/s]
No gold birth places provided; returning (0,0)
Predictions written to vanilla.nopretrain.test.predictions; no targets provided
```

Code 1: vanilla.model model's accuracy for the dev and test set.

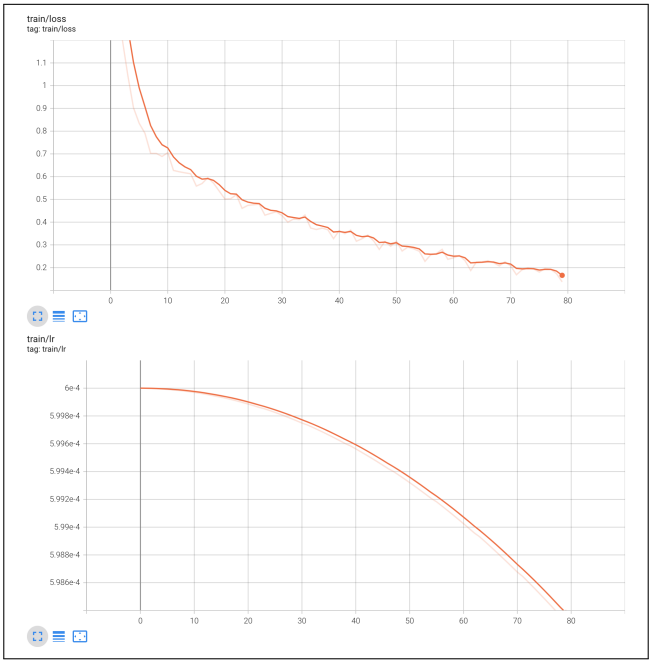

Figure 1: Tensorboard results of the vanilla.model model.

Reference (when only predict "London" as the birth place for everyone in the dev set):

```
python ./src/london_baseline.py
```

Code 2: Command to show the result of the reference.

```
Correct: 25 out of 500: 5.0%
```

Code 3: Reference result.

1.5 (e, 20 points) Define a *span corruption* function for pretraining.

No answers for this part

1.6 (f, 20 points) Pretrain, finetune, and make predictions. Budget 2 hours for training.

Model's accuracy on the dev set:

```
data has 418352 characters, 256 unique.
number of parameters: 3323392
500it [00:11, 45.22it/s]
Correct: 100.0 out of 500.0: 20.0%
data has 418352 characters, 256 unique.
number of parameters: 3323392
437it [00:09, 45.83it/s]
No gold birth places provided; returning (0,0)
Predictions written to vanilla.pretrain.test.predictions; no targets provided
```

Code 4: vanilla.finetune model's accuracy for the dev and test set.

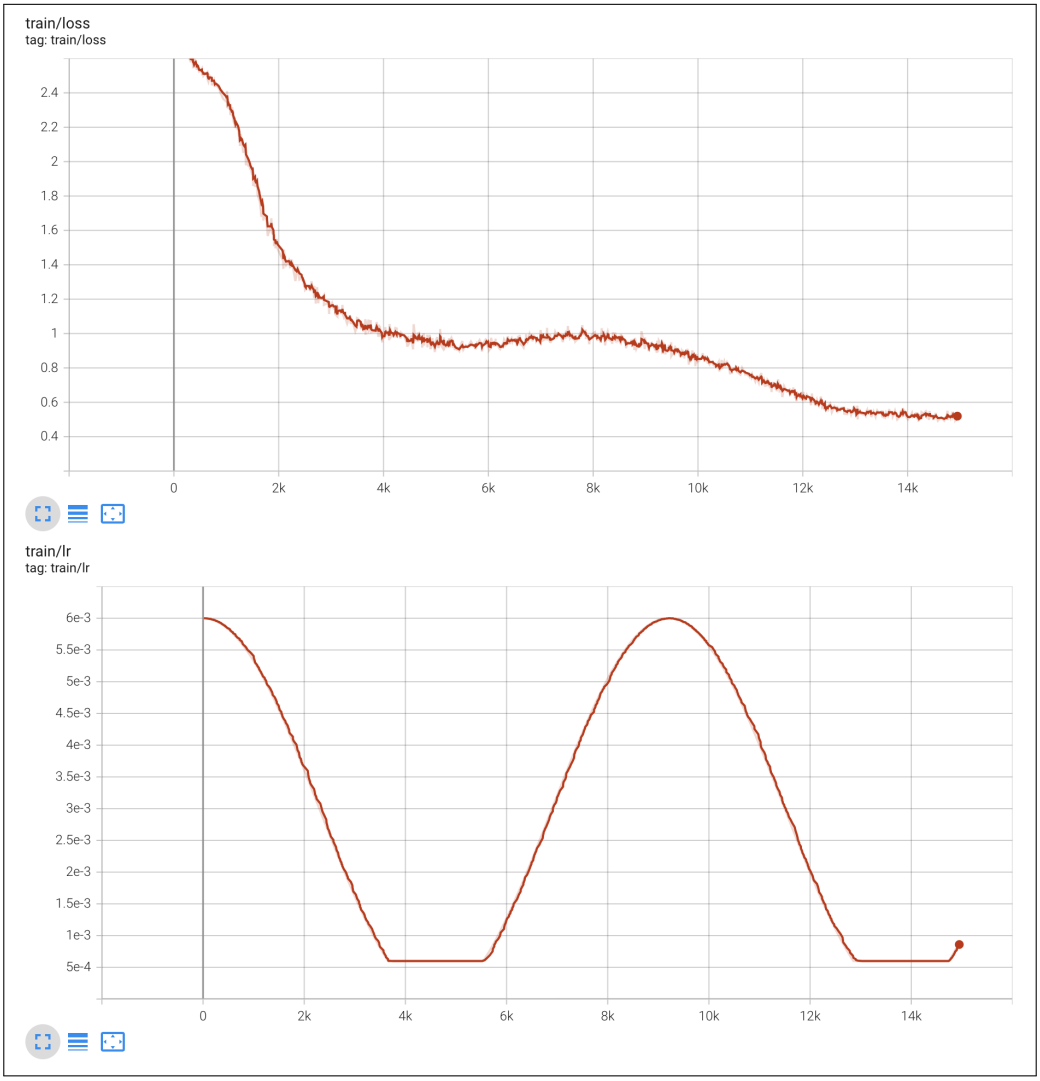

Figure 2: Tensorboard results of the vanilla.finetune model.

1.7 (g, 20 points) Research! Write and try out a more efficient variant of Attention (Budget 2 hours for pretraining!).

1.7.1 (16 points) We'll score your model as to whether it gets at least 5% accuracy on the test set, which has answers held out.

Perceiver attention model's accuracy on the dev set and test set:

```
data has 418352 characters, 256 unique.
number of parameters: 3339776
500it [00:11, 44.95it/s]
Correct: 64.0 out of 500.0: 12.8%
data has 418352 characters, 256 unique.
number of parameters: 3339776
```

```
437it [00:09, 44.13it/s]
No gold birth places provided; returning (0,0)
Predictions written to perceiver.pretrain.test.predictions; no targets provided
```

Code 5: perceiver.finetune model's accuracy for the dev and test set

I can't check the accuracy for the test set by myself because there's no GT for the test set.

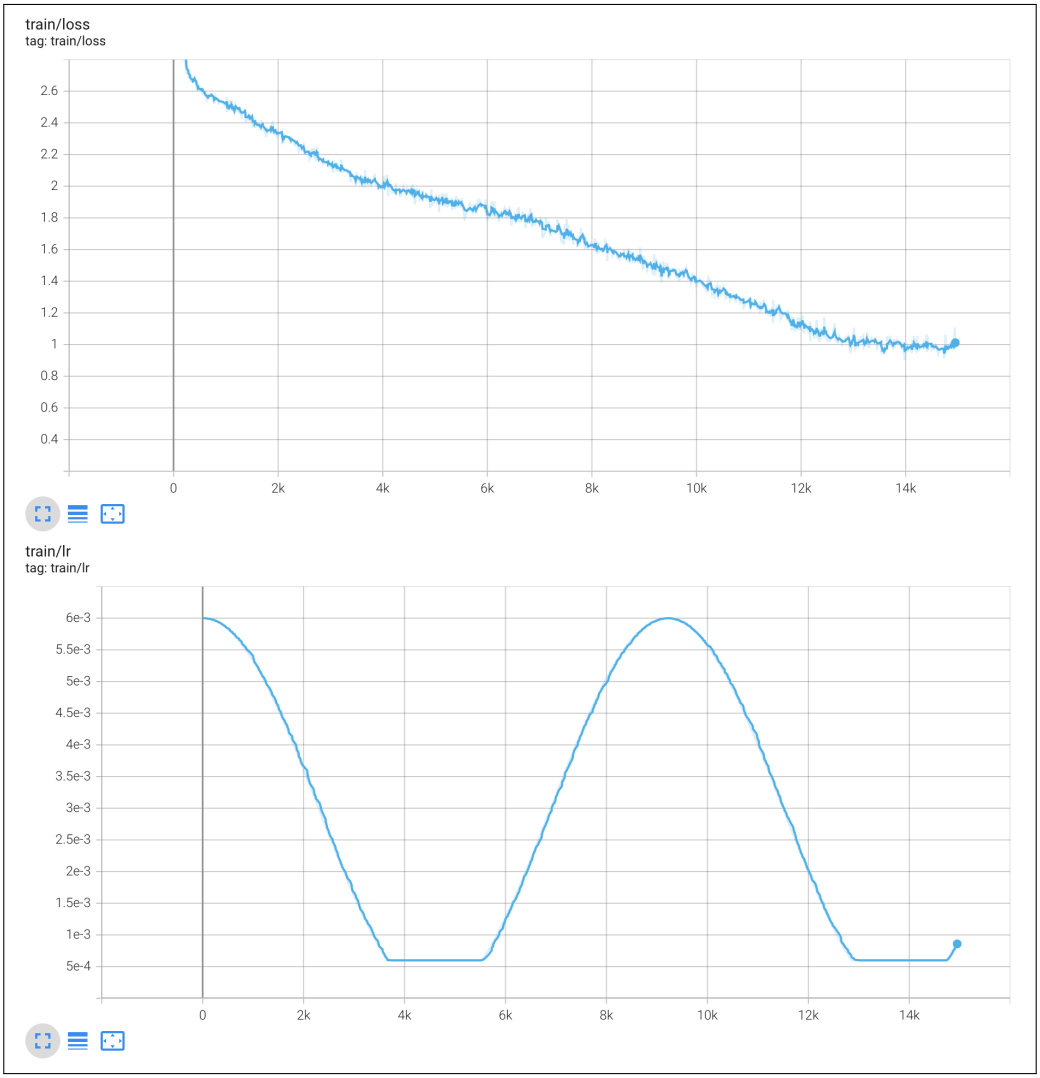

Figure 3: Tensorboard results of the perceiver.finetune model.

1.7.2 (4 points) Provide an expression for the time complexity of the Perceiver model and the vanilla model, in terms of layers (L), input sequence length (l) and basis bottleneck dimension (m).

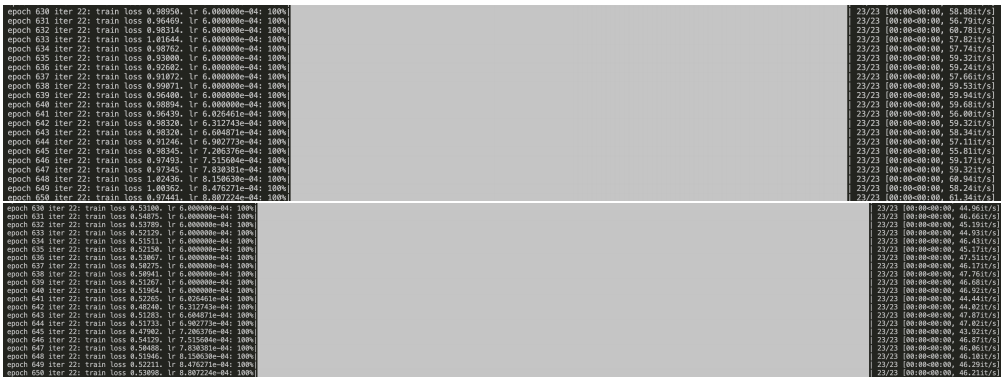

Figure 4: Pre-training process of vanilla model(top) and perceiver model(bottom).

As in Fig. 3, training speed of perceiver model is approximately 30% faster than of vanilla model.

Main difference of the computational cost between vanilla model and Perceiver model is come from the self-attention, which is implemented as `CausalSelfAttention`.

In the vanilla model, the complexity for each layer is $O(l^2)$, as each of the l elements computes attention with respect to l elements. And there are total L layers, so complexity becomes $O(L \cdot l^2)$. On the other hand, in the case of Perceiver model, the self-attention is applied to the projected sequence which has reduced dimension, m, by the first layer(`DownProjectBlock`). The complexity in the first and last layers is still $O(l^2)$ because they include the full sequence. However, for the intermediate layers, the complexity is reduced to $O(m^2)$ where $m < l$. Hence, the total complexity becomes $O(2 \cdot l^2 + (L-2) \cdot m^2)$.

2 Considerations in pretrained knowledge (10 points)

2.1 (a, 2 points) Succinctly explain why the pretrained (vanilla) model was able to achieve an accuracy of above 10%, whereas the non-pretrained model was not.

By pretraining on the `CharCorruption`, the model can understand language generally. It can improve the model by serving as parameter initialization during finetuning on the `NameDataset`. On the other hand, non-pretrained model lacks the foundational knowledge and language understanding compared to the pretrained one.

2.2 (b, 4 points) Take a look at some of the correct predictions of the pretrain+finetuned vanilla model, as well as some of the errors. We think you'll find that it's impossible to tell, just looking at the output, whether the model *retrieved* the correct birth place, or *made up* an incorrect birth place. Consider the implications of this for user-facing systems that involve pretrained NLP components. Come up with two distinct reasons why this model behavior(i.e., unable to tell whether it's retrieved or made up) may cause concern for such applications, and an example for each reason.

There are two distinct reason of "False Positives", incorrect information presented as correct, and "False Negatives", correct information dismissed as incorrect.

If a model frequently hallucinates information, users may accept false information as true. This can lead to the spread of misinformation and people think the system is not trustworthy over time. For example, in recently, there is a popular youtube channel "Psick Univ" which offers various comedy contents. At the beginning of "Narak Quiz Show", one of their popular contents, they introduce the participant by asking who is he(or she) to the ChatGPT with his(or her) name. Of course ChatGPT doesn't know who is he(or she), but it hallucinates as it knows him even with some historical stories. Hosts of the quiz show use it as one of funny situations, but people can think ChatGPT is not trustworthy.

Also, in cases where the model retrieves the correct information but users have no way of knowing if it's accurate or counterfeit, they might ignore correct information. For instance, in a medical context, if a doctor uses an NLP system to query the typical symptoms of a rare disease and the system correctly retrieves the information, the doctor can doubt the accuracy of the system. It can lead to delays in diagnosis or even misdiagnosis, affecting patient care.

2.3 (c, 4 points) If your model didn't see a person's name at pretraining time, and that person was not seen at fine-tuning time either, it is not possible for it to have "learned" where they lived. Yet, your model will produce *something* as a predicted birth place for that person's name if asked. Concisely describe a strategy your model might take for predicting a birth place for that person's name, and one reason why this should cause concern for the use of such applications. (You do not need to submit the same answer for 3c as for 3b.)

When faced with a query that has not encountered during pretraining and fine-tuning, the model may depends on the patterns of given data. It analyze the structure(in this case, the association among characters) and make a guess based on similar names it has encountered. Such strategy can lead to the reinforcement of stereotypes(model associates certain types of names with specific regions). In the end, this approach can lead to inaccuracies.