Deep Learning

Homework 2: Learning on Sequence Data

Ge Congqin, Department of Chemical Engineering, 2020311193

November 20, 2022

1 Part One: RNN

1. **RNN for Language Modeling**: Construct a kind of RNN with GRU block and train your model from scratch on the `train.txt`. Validate your model on the `valid.txt`, and report the training and validation curves.

Solution:

The RNN model is implemented in the `RNN` class in the file `model.py` using `torch.nn.GRU`.

We construct a single-layer model with input size 256 and hidden size 512. The initial learning rate is 0.001 and is multiplied by 0.3 every 20 epochs, and the training batch size is 32. The curves for the loss function, top-1 accuracy and top-10 accuracy over 50 training epochs is given below:

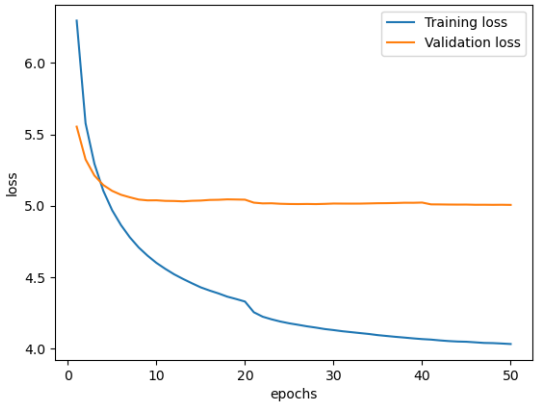


Figure 1: Loss curve for the RNN model

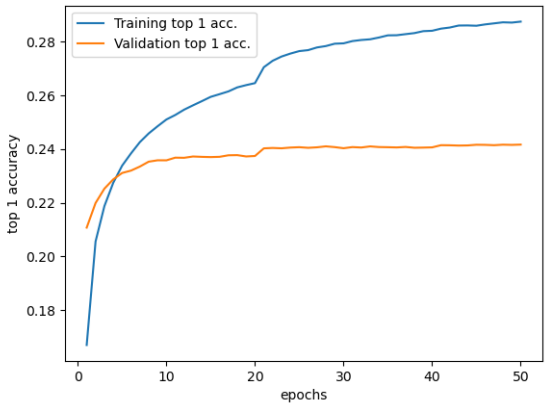


Figure 2: Top-1 accuracy curve for the RNN model

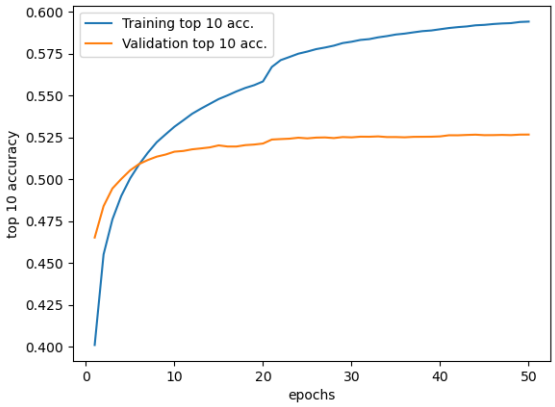


Figure 3: Top-10 accuracy curve for the RNN model

The minimum validation loss is 5.0076, the maximum validation top-1 accuracy is 0.2417, and the maximum validation top-10 accuracy is 0.5267.

2. **LSTM implementation**: Construct a kind of RNN with LSTM block and train your model from scratch (usage of an existing LSTM implementation **is not allowed**). Validate your model, and report the training and validation curves.

Solution:

We build a single-layer LSTM model according to [1]. The LSTM model has similar hyperparameters with the aforementioned RNN model, with input size 256 and hidden size 512. The learning rate strategies used are also the same. We list the curves as follows:

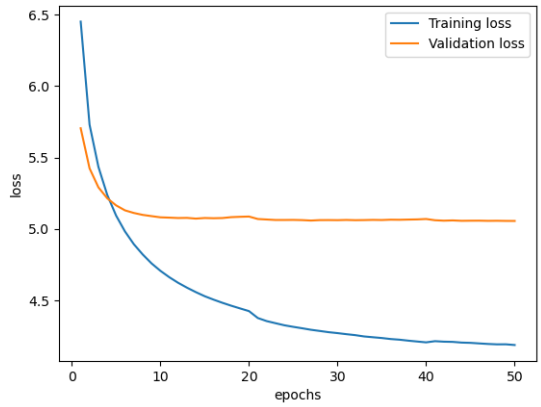


Figure 4: Loss curve for the LSTM model

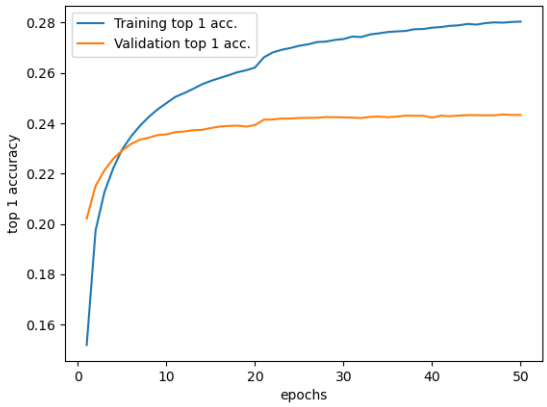


Figure 5: Top-1 accuracy curve for the LSTM model

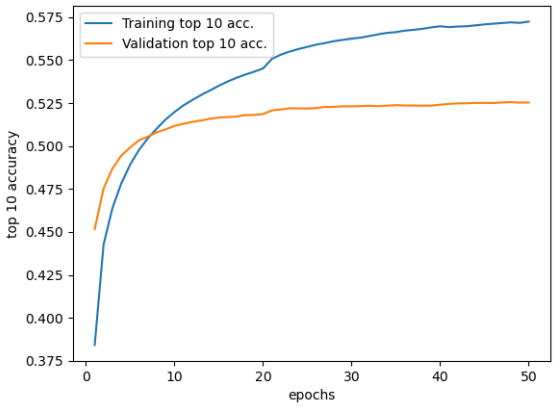


Figure 6: Top-10 accuracy curve for the LSTM model

The minimum validation loss is 5.0566, the maximum validation top-1 accuracy is 0.2435, and the maximum validation top-10 accuracy is 0.5256.

3. **Question answering**: Can you give one advantage and disadvantage of a language model that uses a character-level RNN?

Solution:

Character-level RNN language models (LMs) are better dealing with languages with a rich morphology such as Finish, Turkish and Russian and can mimic grammatically correct sequences for a wide range of languages[2]. However, character-level LMs are weaker at capturing long-distance dependencies and require much bigger hidden layers to successfully model long-term dependencies[2, 3]. Also, it takes a longer path to propagate the error from the softmax at the last time step back to the beginning, so character-level LMs are harder to train[3].

2 Part Two: Transformer

1. **Question answering**: Explain why the following choices are beneficial: (1) Using multiple attention heads instead of one. (2) Dividing by $\sqrt{d/h}$ before applying the softmax function, where d is the feature dimension and h is the number of heads.

Solution:

(1) Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions[4]. The multi-head mechanism works similarly as the different channels in CNNs.

(2) The division operation eliminates the influence of feature dimension and normalizes the variance of $q^\top k$ to 1, so that no large value will be passed into the saturation zone of the Softmax function.

2. **Multihead Attention**: Construct the multi-head attention module based on the given equations. We provide the start code and checking code in `"./src/mha.py"`.

Solution:

The implementations are done in the file `mha.py`. The reported output errors are 1.33×10^{-4} for self attention, 1.40×10^{-4} for masked self attention, and 1.34×10^{-4} for attention output.

3. **Tranformer for Language Modeling**: Construct a Transformer for language modeling. Validate your model, and report training and validation curves.

Solution:

We build our Transformer-based LM according to [5].

Our Transformer LM has 6 encoder layers, with input shape 256 and hidden shape 768. The initial learning rate is 0.0005 and is multiplied by 0.95 every epoch. The training batch size is set as 128. We present the relevent curves below:

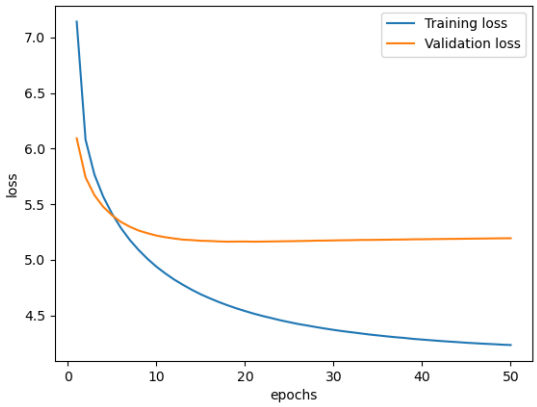


Figure 7: Loss curve for the Transformer model

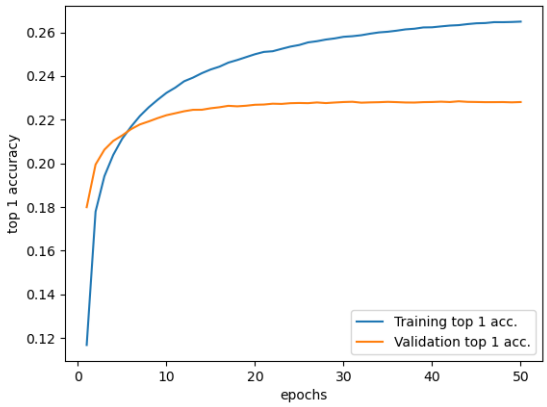


Figure 8: Top-1 accuracy curve for the Transformer model

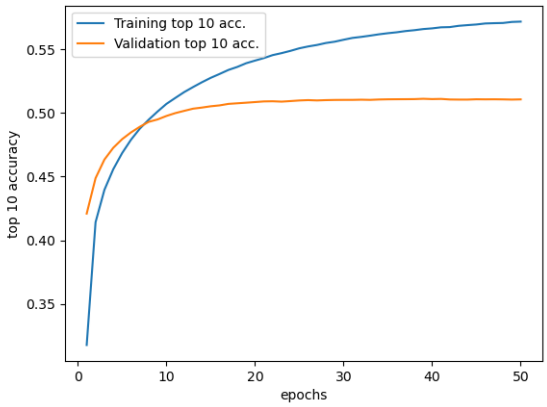


Figure 9: Top-10 accuracy curve for the Transformer model

The minimum validation loss is 5.1638, the maximum validation top-1 accuracy is 0.2284, and the maximum validation top-10 accuracy is 0.5111.

4. **Masking**: As we have learned in class, masking is used in Transformer to keep it autoregressive. Explain why masking is necessary for our task and how it is implemented in your code.

Solution:

Masking is necessary for prevention the attention mechanism of a transformer from "cheating" in the decoder when training[6]. The attention mechanism should only be visible to the previous tokens when predicting the next token, and the mask is used to set future positions to `-inf` before the softmax step.

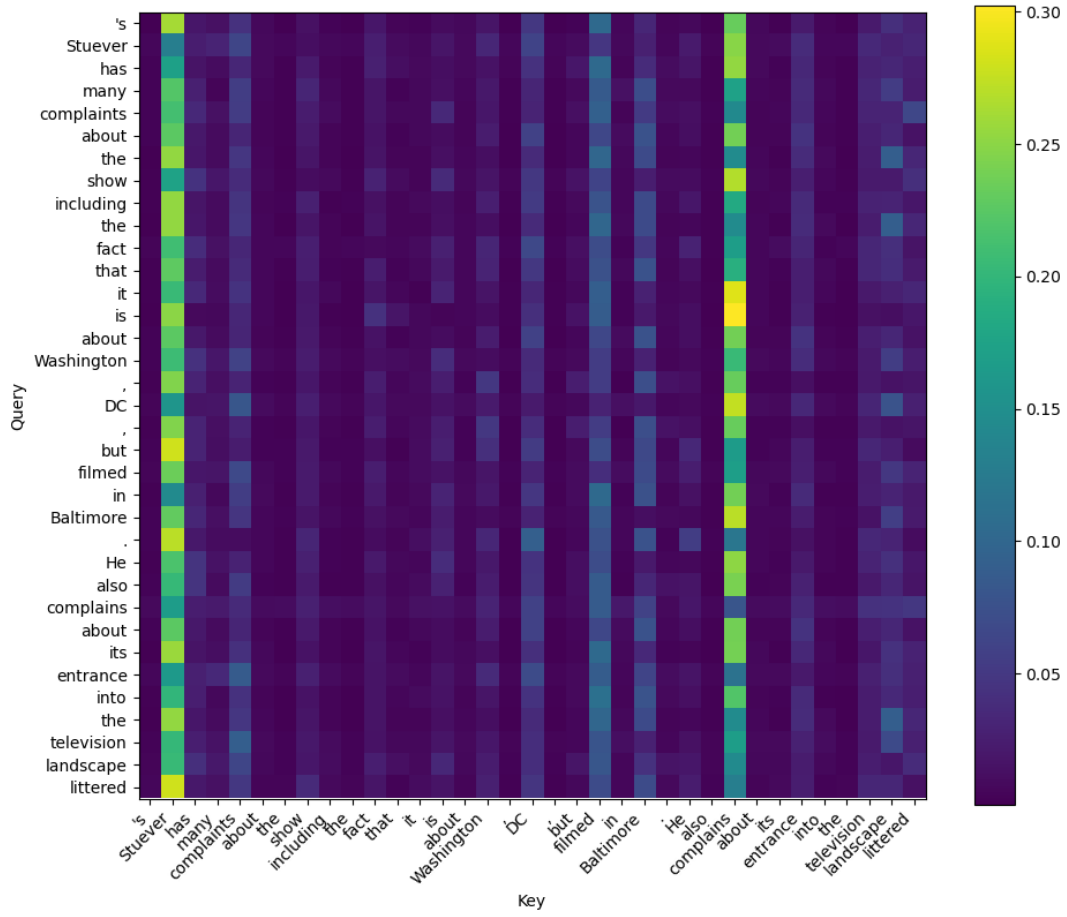
We implement masking through the following lines in our code:

```
mask = torch.triu(torch.ones(sql_len, sql_len) * float("-inf"), diagonal =
    1).to(input.device)
...
output = self.transformer_encoder(embeddings, mask)
```

5. **Attention Visualization**: Visualize the attention weights of some typcial words and check whether the Transformer can learn meaningful attention weights.

Solution:

We visualize the weights of the last multihead attention layer in the Transformer model.



We can see that the attention for all queries focuses on the subject of the first sentence and the verb of the second sentence. This may imply that the attention mechanism attempts to extract the basic skeleton of a sentence in order to grasp the context of a sequence.

3 Part Three: Improve your language model

1. Start with how you feel after completing the first few tasks and adopt an extra technique you find in other materials to further improve your language model (no matter for performance or efficiency). Please explain why you choose it, and check whether it works on our task.

Solution:

By completing the previous tasks (RNN, LSTM, Multi-head attention, Transformer), I feel that I have

gained a keener grasp of the basic LM techniques. Previously I have come upon with these techniques multiple times but had no idea how these models are designed or implemented, but thanks to this assignment I have managed to construct my own models.

The extra LM technique I choose is GPT, since it uses multi-head self attention as the basis of the model with some minor modifications. The major novelties of a GPT block is the inclusion of layer normalization and shortcut connections. The implementation of the GPT model is in class `model.GPT`, based on the tutorial in [7].

Due to sigificant overfitting, the number of epochs is reduced to 30, the initial learning rate is changed to 0.0001 and is multiplied by 0.95 every epoch. The relevant curves are presented below.

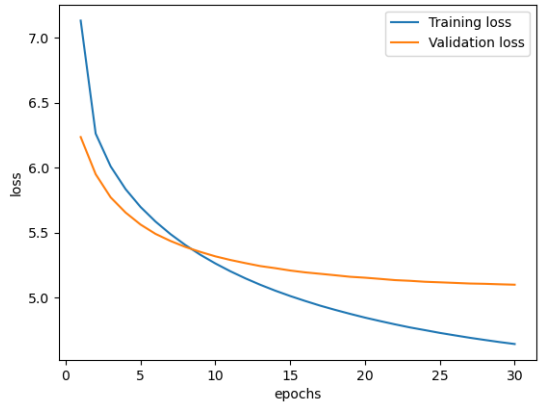


Figure 10: Loss curve for the GPT model

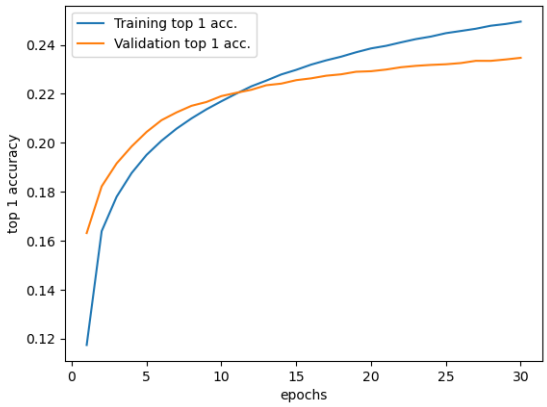


Figure 11: Top-1 accuracy curve for the GPT model

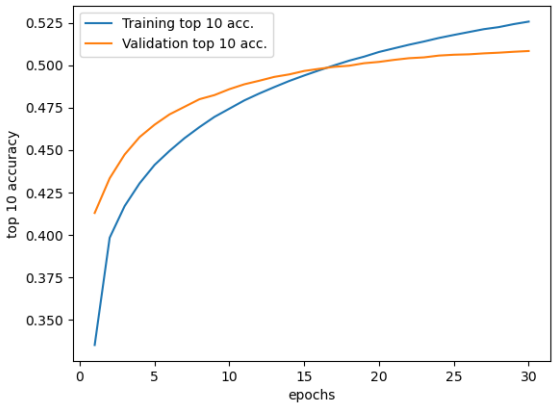


Figure 12: Top-10 accuracy curve for the GPT model

The minimum validation loss is 5.0984, the maximum validation top-1 accuracy is 0.2347, and the maximum validation top-10 accuracy is 0.5083. No visible performance improvement is observed as we move to the GPT model, possibly because GPT relies heavily on the pretraining (with huge datasets) and the fine-tuning steps, while we are only training on a relatively smaller dataset and corpus.

References

[1] Piero Esposito. Building a LSTM by hand on PyTorch. Accessed 9 November 2022. https://towardsdatascience.com/building-a-lstm-by-hand-on-pytorch-59c02a4ec091

[2] Data Science Stack Exchange. What is the difference between word-based and char-based text generation RNNs?. Accessed 20 November 2022. https://datascience.stackexchange.com/questions/13138/what-is-the-difference-between-word-based-and-char-based-text-generation-rnns

[3] Quora. What are the drawbacks of character-level RNN, compared with word-level RNN?. Accessed 20 November 2022. https://www.quora.com/What-are-the-drawbacks-of-character-level-RNN-compared-with-word-level-RNN

[4] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[5] PyTorch Tutorials. Sequence-to-Sequence Modeling with nn.Transformer and TorchText. Accessed 13 November 2022. https://torchtutorialstaging.z5.web.core.windows.net/beginner/transformer_tutorial.html

[6] Samuel Kierszbaum. Masking in Transformer's self-attention mechanism. Accessed 20 November 2022. https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

[7] Jake Tae. GPT from Scratch. Accessed 27 November 2022. https://jaketae.github.io/study/gpt/