Large Language Models (Homework 1)

Due date : 2025/10/17 23:55:00 (Hard Deadline)

1 N-Gram, RNN and LSTM Language Models (70%)

In this section, you need to implement n-gram, recurrent neural network (RNN) and long short term memory (LSTM) language models for "next token" prediction. The dataset used is sampled from the "Recipes" of the "Food.com Recipes and Interactions" dataset, which is available on Kaggle. This dataset includes step, ingredients, and some related descriptions. In this exercise, you will only use the "steps" portion for model training and prediction.

Dataset description:

- This dataset is divided into a training set (train.txt) with approximately 2600K entries and a test set (test.txt) with approximately 650K entries.
- You must use train.txt to train the model and use test.txt to evaluate the model accuracy of test data.
- Each entry contains a sentence with at least two words describing a recipe step.
- In the implementation, you will need to split the sentence in an entry into words, and create a vocabulary for the words in your dataset.
- The data samples have been shuffled, you do not need to reshuffle it.
- The file incomplete.txt contains 10 incomplete sentences. You use the trained language models to complete it.

Please follow the steps below to implement your program:

1. Calculate the following n-gram language model where $C(\cdot)$ denotes the number of occurrences. You can use defaultdict() in Python. After training, please evaluate your model by using the test set and calculate the prediction accuracy (%) in test.txt and complete sentences in incomplete.txt.

$$p(w_t \mid w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}) = \frac{C(w_{t-n+1}, w_{t-n+2}, \ldots, w_t)}{C(w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1})}$$

   (a) Use $n = 2$ and $n = 3$ to calculate bigrams and trigrams, respectively, and evaluate the test accuracy in test.txt and make some discussion. (10%)

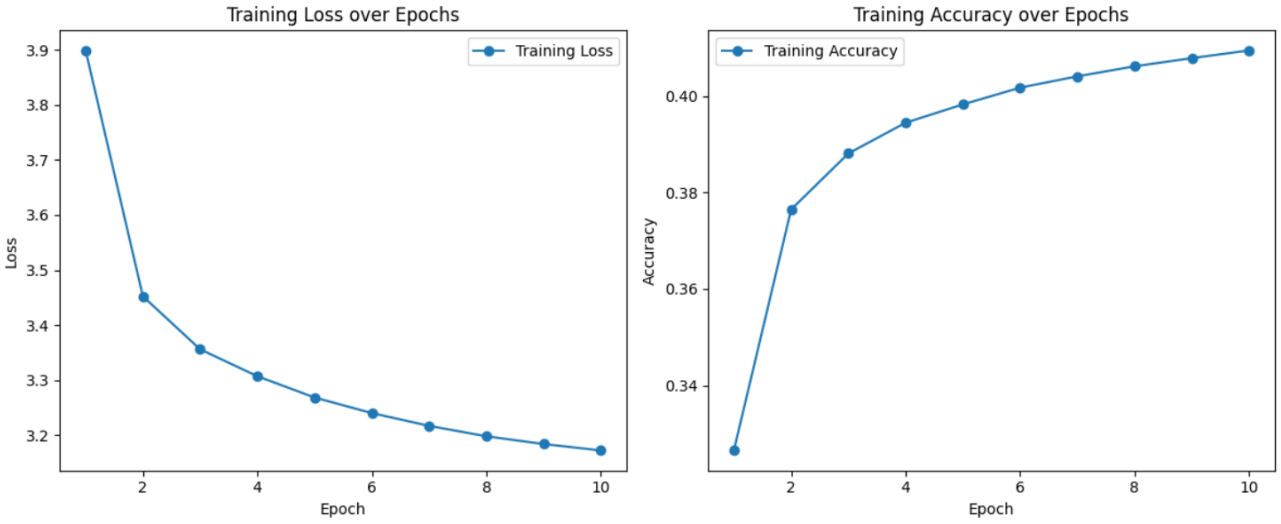
   (b) Please observe the hardware usage in your computer while the models are running, and make some discussion. (5%)

   (c) Please use the test sentences from incomplete.txt to evaluate the model. This test set contains some incomplete sentences. Use the trained trigram model to complete the sentences (to reach a length of 20 words). (5%)

2. You are allowed to use the PyTorch library to implement an Recurrent Neural Network (RNN) model. You must create a vocabulary and implement the conversion between words and indices. Use cross-entropy as the loss function and Adam as the optimizer. The hyperparameter settings for the model are suggested as follows:

   - hidden dimension: 128
   - number of layers : 2
   - learning rate : 0.001
   - number of epochs : 10
   - batch size : 32

   (a) Plot the learning curves (training loss) and the accuracy rate curves of training data. (10%)



NOTE: Figure above shows an example. The result might be different.

   (b) Evaluate the RNN model on test.txt and report accuracy and perplexity. (5%)

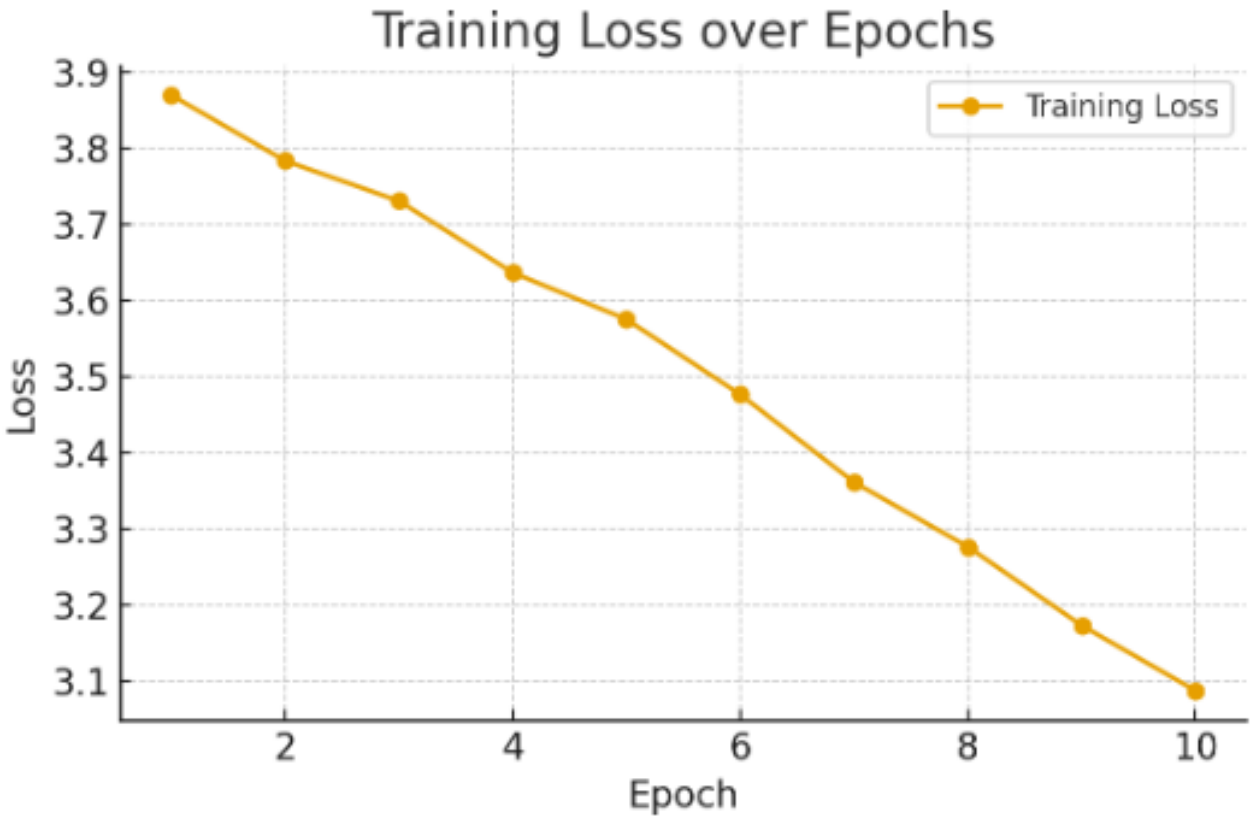
$$H = -\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

$$\text{Perplexity} = 2^H$$

(c) Please use the test sentences from incomplete.txt to evaluate the model. This test set contains some incomplete sentences. Use the trained RNN model to complete the sentences (to reach a length of 20 words). (5%)

3. You are also required to implement a Long Short-Term Memory (LSTM) model as an extension of the RNN. The model should follow the same settings as the RNN implementation, including vocabulary construction, word-to-index mapping, loss function, optimizer, and suggested hyperparameters.

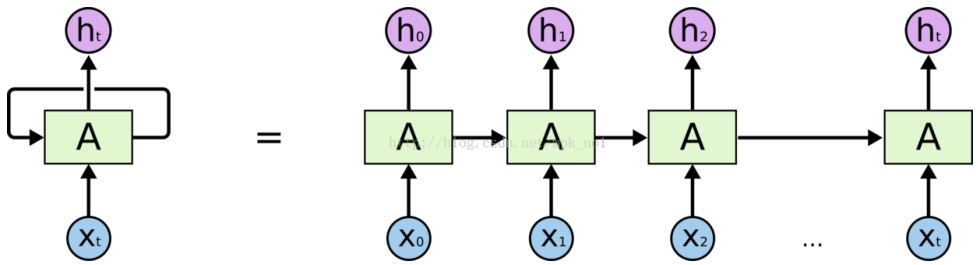
   (a) Plot the learning curves (training loss) and the accuracy rate curves of training data. (10%)



   (b) Use test.txt as input to your LSTM model and calculate its accuracy and perplexity. (5%)

   (c) Use the test sentences from incomplete.txt to evaluate the model. This test set contains some incomplete sentences. Use the trained LSTM model to complete the sentences (to reach a length of 20 words). (5%)

4. Analyze and compare the differences among the n-gram model, the RNN model, and the LSTM model, including the hardware resources required for their operations, the computation time taken for training, and the results of perplexity and accuracy. (10%)



NOTE:

- The accuracy for predicting the next token is defined as: $\frac{\text{the number of correct predictions}}{\text{the total number of predictions}}$
- The accuracy will likely range between 20% and 40%.

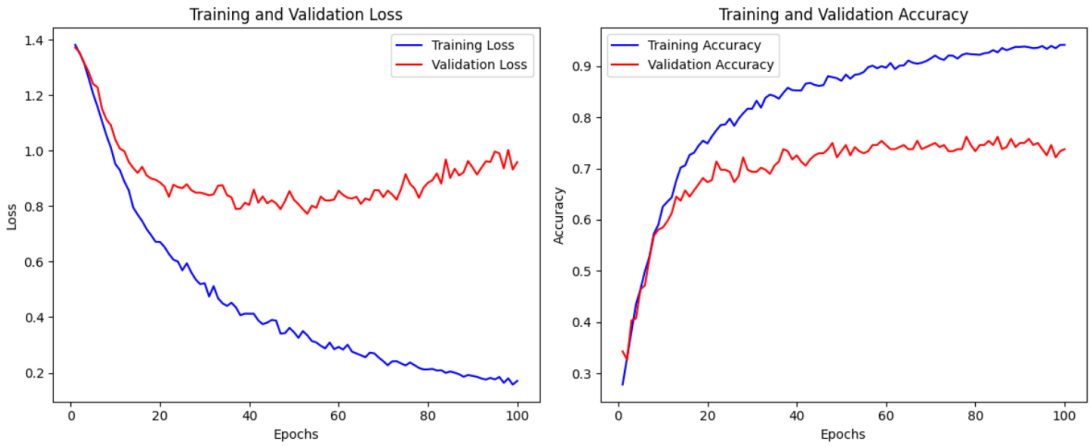
2 Self-Attention Layer in Transformer (30%)

You will be provided with an news classification dataset, which is sampled from the "AG News Classification" dataset on Kaggle. You will also receive a partially completed code. Please follow the instructions to complete the code and train a model to perform a classification task.

Dataset description:

- The dataset is divided into a training set (train.csv) with 3000 entries, a validation set (val.csv) with 250 entries and a test set (test.csv) with 2000 entries.
- You must use train.csv to train the model and use test.csv to evaluate the model performance.
- Each entry contains the title, description and its corresponding news label. There are four categories in total.
- The data samples have been shuffled. You do not need to reshuffle it.

1. Please complete the code in the atten.ipynb file, which is an encoder for a text classification task. You only need to write the Multi-head Attention and Transformer Encoder layer, which correspond to the "TODO" section in the code. Do not use the class torch.nn.MultiheadAttention, and use the provided hyperparameters.
   (a) Please set the number of attention heads to 4 and plot the learning curves (training loss) and the accuracy rate curves of training and validation data. (15%)



NOTE: Figure above shows an example. The result might be different.

(b) Use test.csv as input to your model and calculate the test accuracy. (5%)

(c) Reduce the number of attention heads (down to 1) and increase it (up to 8). Compare the results and provide some discussion. (10%)

3 Rule

- In your submission, you need to submit two files. And only the following file format is accepted:
  - **hw1_<ProblemNumber>_<StudentID>.ipynb** file which need to contain all the results, codes and reports for each exercise (e.g. **hw1_2_0123456.ipynb**).
- Implementation will be graded by
  - Completeness
  - Algorithm correctness
  - Description of model design
  - Discussion and analysis
- Only Python implementation is acceptable.
- For n-gram problem, you can only use basic libraries, such as NumPy, Pandas, Random, etc. Advanced libraries such as scikit-learn and PyTorch are forbidden, you should implement n-gram by yourself.
- For RNN and attention problem, you can use PyTorch to implement the model, except torch.nn.MultiheadAttention.
- You need to use the GPU for RNN and attention problem.
- DO NOT PLAGIARIZE. (We will check program similarity score.)