ECE 661: Homework #3
Understand and Implement Recurrent Models and Transformers

Hai Li

ECE Department, Duke University — September 9, 2021

Objectives

Homework #3 covers the contents of Lectures 09∼10. This assignment includes basic knowledge about RNNs and Transformers, detailed instructions on how to implement the core of the models and the training pipeline for training sentiment analysis on the IMDB dataset, how to improve the training pipeline, and how to use a pre-trained tokenizer. In this assignment, you will gain hands-on experience in training an LSTM and a Transformer on the IMDB dataset, while also learning techniques for improving the performance of your models.

Some parts of this assignment require advanced features in the latest PyTorch version, which are unfortunately not available on our JupyterLab server. Thus **using Google CoLab is strongly encouraged** for finishing the lab questions. A brief introduction on setting up and using CoLab is provided at the beginning of Lab 1. When conducting the lab projects, actively referring to the **NumPy/PyTorch tutorial** slides on Sakai for the environment setup and instructions for NumPy/PyTorch utilities can be very helpful.

Warning: You are asked to complete the assignment independently.

This lab has 100 points plus 10 bonus points, yet your final score cannot exceed 100 points. The submission deadline will be **11:59pm, Tuesday, October 12**. For each lab, we provide one or multiple notebooks for you to start with. Your task is to fill in the missing modules in these notebooks. No new notebook or scripts need to be created. You will need to submit two independent files including:

1. A self-contained PDF report, which provides answers to all the conceptual questions and clearly demonstrates all your lab results and observations. Remember, **do NOT generate PDF from your Jupiter notebook to serve as the report**, which can increase the TA's burden of grading. Besides, you pdf file should also contain the plotted figures (instead of submit separately). You can upload a saved figure or a screenshot of the figure to the pdf report.
2. `code.zip`, a zipped code file which contains 4 Jupiter notebooks `lab1.ipynb`, `lab2.ipynb`, `transformer.ipynb`, and `attention.ipynb`.

Note that 20 percent of the grade will be deducted if the submissions doesn't follow the above guidance.

1 True/False Questions (30 pts)

For each question, please provide a short explanation to support your judgment.

Problem 1.1 (3 pts) In the self-attention layer of Transformer models, we compute three core variables—key, value and query.

Problem 1.2 (3 pts) In the self-attention layer of Transformer models, the attention is denoted by the cosine similarity between the key and value.

Problem 1.3 (3 pts) In the self-attention layer of Transformer models, after obtaining the attention matrix, we need to further apply a normalization on it (e.g., layer normalization or batch normalization).

Problem 1.4 (3 pts) The decoder of Transformer learns auto-regressively.

Problem 1.5 (3 pts) BERT's architecture is a Transformer encoder while GPT's architecture is a Transformer decoder.

Problem 1.6 (3 pts) BERT's pre-training objectives include (a) masked token prediction (masked language modeling) and (b) sentence order prediction.

Problem 1.7 (3 pts) GPT is a zero-shot learner (can be transferred to unseen domain and tasks) while BERT is not.

Problem 1.8 (3 pts) Gradient clipping can be used to alleviate gradient vanishing problem.

Problem 1.9 (3 pts) Word embeddings can only contain positive values.

Problem 1.10 (3 pts) The memory cell of an LSTM is computed by a weighted average of previous memory state and current memory state where the sum of weights is 1.

2 Lab 1: Implement and train an LSTM for sentiment analysis (35 pts)

Google colab is a useful tool for modeling training. In order to run `ipynb` document, you need to upload your jupyter notebook document to google drive. Then double click the file in the google drive, which will lead you to google colab. Moreover, it'd be more efficient to use GPU to train your LSTM and Transformer models in this assignment. To use GPU, please check *Runtime* > *Change runtime type*, and select *GPU* as the hardware accelerator. You may click on side bar document icon to upload your dataset and python file there.

In this lab, you will learn to implement an LSTM model for sentiment analysis. **The goal is to reach at least 80% validation and testing accuracy on the IMDB dataset within at most 5 epochs.** Sentiment analysis [1, 2] is a classification task to identify the sentiment of language. It usually classifies data into two to three labels: positive, negative or/and neutral. IMDB [3] is one of the most widely used dataset for binary sentiment classification. It uses only positive and negative labels. IMDB contains 50,000 movie review data collected from popular movie rating service IMDB. You may find more details at https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

Lab 1 (35 points)

To start this assignment, please open the provided Jupyter Notebook `lab1.ipynb`.

(a) (5 pts) Implement your own data loader function. First, read the data from the dataset file on the local disk. Then split the dataset into three sets: train, validation, and test by $7:1:2$ ratio. Finally return `x_train`, `x_valid`, `x_test`, `y_train`, `y_valid` and `y_test`, where `x` represents reviews and `y` represent labels.

(b) (5 pts) Implement the `build_vocab` function to build a vocabulary based on the training corpus. You should first compute the frequency of all the words in the training corpus. Remove the words that are in the `STOP_WORDS`. Then filter the words by their frequency ($\geq$ `min_freq`) and finally generate a corpus variable that contains a list of words.

(c) (5 pts) Implement the `tokenization` function. For each word, find its index in the vocabulary. Return a list of integers that represents the indices of words in the example.

(d) (5 pts) Implement the `__getitem__` function in the `IMDB` class. Given an index i, you should return the i-th review and label. The review is originally a string. Please tokenize it into a sequence of token indices. Use the `max_length` parameter to truncate the sequence so that it contains at most `max_length` tokens. Convert the label string ('positive' / 'negative') to a binary index, such as 'positive' is 1 and 'negative' is 0. Return a dictionary containing three keys: '`ids`', '`length`', '`label`' which represent the list of token ids, the length of the sequence, the binary label.

(e) (10pts) Implement the LSTM model for sentiment analysis.

  (a) (5pts) Write the initialization function. Your task is to create the model by stacking several necessary layers including an embedding layer, an lstm cell, a linear layer, and a dropout layer. You can call functions from Pytorch's nn library. For example, nn.Embedding, nn.LSTM, nn.Linear.

  (b) (5pts) Write the forward function. Decide where to apply dropout. The sequences in the batch have different lengths. Write/call a function to pad the sequences into the same length. Apply a fully-connected (fc) layer to the output of the LSTM layer. Return the output features which is of size [batch size, output dim].

(f) (5pts) Train the model for 5 epochs. Copy the plotted figures of training/validation loss/accuracy to your self-contained pdf report. What is your testing accuracy? (The provided code contains the plot function and computation of test accuracy. You just need to report the value of testing accuracy.)

At the end of Lab 1, you should obtain at least 80% validation and testing accuracy if all the steps are completed properly. Please do not run more than 5 epochs as several epochs suffices to achieve more than 80% accuracy. You are required to submit the completed version of `lab1.ipynb` for Lab 1.

3 Lab 2: Implement and train a Transformer for sentiment analysis (35 pts)

In this lab, we focus on Transformer models. **The goal is to reach at least 80% validation and testing accuracy on the IMDB dataset within at most 5 epochs.**

Lab 2 (35 points)

Please open the notebook called `lab2.ipynb`.

(a) (5 pts) Implement Transformer's multi-head attention mechanism. Open the `attention.py` file and implement the class called `MultiHeadedAttention`. Use the layers initialized in the `__init__` function. Compute attention between the query and key. Apply the obtained attention scores on value. Take care of the multi-head machenism. Apply a fc layer to the output before return. The `transformer.py` file calls the `MultiHeadedAttention` module in the `attention.py` file to establish a Transformer model. Then the `lab2.ipynb` notebook calls the Transformer model when initializing the model for sentiment analysis.

(b) (5 pts) Train the model for 5 epochs. Copy the plotted figures of training/validation loss/accuracy to your self-contained pdf report. What is your testing accuracy? What is your model size? (The provided code contains the plot function and computation of test accuracy. You just need to report the value of testing accuracy.)

(c) (5 pts) Until now, you have been using your own implemented vocabulary and tokenization function. What if you want to use a tokenizer of a pre-trained model such as BERT? Please replace your the vocabulary and tokenizer with `transformers.AutoTokenizer.from_pretrained(transformer_name)`, where `transformer_name = 'bert-base-uncased'`. Note that this will/should not load the pre-trained BERT model but just the tokenizer (which implements the WordPiece algorithm).

(d) (5 pts) Train the model for 5 epochs. Copy the plotted figures of training/validation loss/accuracy to your self-contained pdf report. What is your testing accuracy? What is your model size? What do you observe when comparing this with the original model's performance? (The provided code contains the plot function and computation of test accuracy. You just need to report the value of testing accuracy.)

(e) (5 pts) Replace the current `PositionalEmbedding` class in the `transformer.py` file with a trainable one. The current `PositionalEmbedding` uses the sine and cosine functions to initialize the positional embedding. These functions are deterministic. Could you change the class to use a trainable neural network to initialize the position embedding? For example, you can use the same embedding layer as the token embedding layer.

(f) (5 pts) Train the model with the updated `PositionalEmbedding` for 5 epochs. Copy the plotted figures of training/validation loss/accuracy to your self-contained pdf report. What is your testing accuracy? What is your model size? (The provided code contains the plot function and computation of test accuracy. You just need to report the value of testing accuracy.)

(g) (5 pts) Warm start learning rate schedule is a useful technique for training Transformers. The learning rate starts from a very small value at the beginning of the training. In the first few epochs, it is gradually increased to a peak value. Afterwards, the learning rate gradually decreases throughout the remaining training. Please implement the warm start learning rate schedule and run your code with it. You can choose your own learning rate values and schedule parameters such as the rising and falling speed (slope).

Up to now, you shall have established a Transformer encoder for sentiment analysis. Remember, you are required to submit `lab2.ipynb`, `transformer.py` and `attention.py` for Lab 2.

Info: Additional requirements:

- **DO NOT** copy code directly online or from other classmates. We will check it! The result can be severe if your codes fail to pass our check.

Info: As this assignment requires much computing power of GPUs, we suggest:

- Plan your work in advance and start early. We will **NOT** extend the deadline because of the unavailability of computing resources.
- Be considerate and kill Jupyter Notebook instances when you do not need them.
- **DO NOT** run your program forever. Please follow the recommended/maximum training budget in each lab.

References

[1] P. D. Turney, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," *arXiv preprint cs/0212032*, 2002.

[2] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," *arXiv preprint cs/0409058*, 2004.

[3] S. Dooms, T. De Pessemier, and L. Martens, "Movietweetings: a movie rating dataset collected from twitter," in *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, vol. 2013, p. 43, 2013.