

Homework 5

General Instructions

This homework must be turned in on both Gradescope and Brightspace by 11:59 pm on the due date. It must be your own work and your own work only—you must not copy anyone's work, or allow anyone to copy yours. This extends to writing code. You may consult with others, but when you write up, you must do so alone. Your homework submission must be written and submitted using Jupyter Notebook (.ipynb). **No handwritten solutions will be accepted.** You should submit:

1. On **Gradescope**: one Jupyter Notebook(.ipynb) containing your solutions for each Problem in this assignment. Each Jupyter Notebook should be named as $(netid)_{hw}(\#Homework)_p(\#Problem).ipynb$. For example, `bz2058_hw5_p1.ipynb`.
2. On **Brightspace**: one .zip file containing all the notebooks, datasets, and other supporting materials involved in this homework. The .zip file should be named as $(netid)_{hw}(\#Homework).zip$. For example, `bz2058_hw5.zip`.

Please make sure your answers are clearly structured in the Jupyter Notebooks:

1. Copy the Template Notebooks provided on Brightspace and write your solutions there.
2. Label each question part clearly. Do not include written answers as code comments. The code used to obtain the answer for each question part should accompany the written answer.
3. All plots should include informative axis labels and legends. All codes should be accompanied by informative comments. All output of the code should be retained.
4. Math formulas can be typesetted in Markdown in the same way as \LaTeX . A [Markdown Guide](#) is provided on Brightspace for reference.

For more homework-related policies, please refer to the syllabus.

Problem 1 - *Sentiment Analysis using recurrent models* 15 points

In this problem, you will compare the performance of RNN, LSTM, GRU, and BiLSTM for the task of sentiment analysis. You'll use the IMDB sentiment analysis dataset referenced below for this task. For each model, use a single cell, and keep the number of units fixed to 256. Train each model for 10 epochs using the Adam optimizer, batch size of 256, and a learning rate of 0.01.

1. Import the dataset and convert it into vector form using the Bag of Words technique. (1.5)
2. Define an RNN model and train it on the dataset. (3)
3. Define an LSTM model and train it on the dataset. (3)
4. Define a GRU model and train it on the dataset. (3)
5. Define a BiLSTM model and train it on the dataset. (3)
6. Compare the performance of all the models. In which case do you get the best accuracy? (1.5)

References:

- IMDB Dataset of 50K Movie Reviews.
Available at <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>

Homework 5

Problem 2 - *Training a simple chatbot using a seq-to-seq model* 15 points

We will train a simple chatbot using movie scripts from the Cornell Movie Dialogs Corpus based on the **PyTorch Chatbot Tutorial** referenced below. This tutorial allows you to train a recurrent sequence-to-sequence model. You will learn the following concepts:

- Handle loading and pre-processing of the **Cornell Movie-Dialogs Corpus** dataset referenced below
- Implement a sequence-to-sequence model with **Luong attention mechanism(s)** referenced below
- Jointly train encoder and decoder models using mini-batches
- Implement greedy-search decoding module
- Interact with the trained chatbot

We will use the code in the tutorial as the starting code for this problem:

1. Make a copy of the notebook of the **PyTorch Chatbot Tutorial**, follow the instructions to train and evaluate the chatbot model in your local environment. **(3)**
2. Watch the video tutorial referenced below and learn how to use **Weights and Biases (W&B)** to run a hyperparameter sweep. You will instrument the notebook to use W&B to run some hyperparameter sweeps in the next steps.
3. Create a sweep configuration using the **W&B Random Search** strategy for the following hyperparameters: **(5)**
 - Learning rate: [0.0001, 0.00025, 0.0005, 0.001]
 - Optimizer: [adam, sgd]
 - Clip: [0, 25, 50, 100]
 - teacher_forcing_ratio: [0, 0.5, 1.0]
 - decoder_learning_ratio: [1.0, 3.0, 5.0, 10.0]
4. Run your hyperparameter sweeps using GPU-enabled environment and observe the results in the W&B console. **(3)**
5. Extract the values of the hyperparameters that give the best results (minimum loss of the trained model). Explain which hyperparameters affect the model convergence. Use the feature importance of W&B to help your analysis. Save the trained model that had the lowest loss. **(4)**

References:

- The Cornell Movie Dialogs Corpus dataset.
Available at https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html
- PyTorch Chatbot Tutorial.
Available at https://pytorch.org/tutorials/beginner/chatbot_tutorial.html
- Luong et al. Effective Approaches to Attention-based Neural Machine Translation.
Available at <https://arxiv.org/abs/1508.04025>

Homework 5

- Hyperparameter sweeps with Weights and Biases Framework tutorial.
Video available at <https://www.youtube.com/watch?v=9zrmUIIScdY>
Notebook available at https://colab.research.google.com/github/wandb/examples/blob/master/colabs/pytorch/Organizing_Hyperparameter_Sweeps_in_PyTorch_with_W%26B.ipynb
- Weights and Biases Website.
Available at <https://wandb.ai/site>

Problem 3 - *Attention in Transformer* 10 points

1. From each of the encoder's input vectors (e.g., the embedding of each word), how many vectors are derived in a self-attention block in a Transformer? What are these vectors called? **(2)**
2. In self-attention, how do we calculate the softmax scores for attention using the vectors in the previous part? Explain. **(2)**
3. Multi-headed attention gives the attention layer multiple "representation subspaces". If we have 8 heads in a self-attention block each with input vectors of size 512 and output vectors of size 512, how many weight matrices do we need to learn in total across these 8 heads? What is the size of these matrices for an input of size 4 word embeddings? **(3)**
4. The feed-forward layer following the self-attention is expecting a single matrix (a vector for each word). How can we go from the output of multiple heads to a single matrix input for the feed-forward layer? **(3)**

Problem 4 - *Using BERT for Question Answering* 10 points

Please find this problem in the [template](#) provided on Brightspace.