AI51701/CSE71001 Assignment 3

Neural Machine Translation

Due: June 5, 2022, 11:59:00pm KST

Submission Instructions: You shall submit this assignment on BlackBoard as two submission files – your code as `assignment3coding.zip` and write up for `assignment3.pdf`; Run the `collect_submission.sh` script to produce your `assignment3.zip` file. This time, we use Jupyter Notebook and Google Colab. It is your responsibility to make sure your code is runnable at Google Colab. If your code is not runnable, you will get no point. Also do not write down your name/student ID in your submission.

Collaboration Policy: You are welcome to discuss assignments with others in the course, but solutions and code must be written individually.

For the following coding problems, please use the provided code for data preprocessing.

1 Implementing sequence-to-sequence model with attention (100 points)

In this problem, we implement sequence-to-sequence model with attention as learned in our class to build a Neural Machine Translation (NMT) system. We provide jupyter notebook file (`assn3_1_seq2-seq_attention.ipynb`) for implementing the model. For decode and encoder models, we use unidirectional LSTMs. For computing attention weights, we use the basic dot-product attention that we also learned in our class.

(a) Implement encoder, decoder, and training and inference code as instructed in comments. Report your translation results for a few examples in the validation set and attention visualization of them. Report BLUE score for the test set (80 points).
(b) Tune your model and code to get the best performance you can get, and report BLUE score for the test set. Explain why your setting achieves this (10 points). Note that training of your model should be done in reasonable time at Google Colab (10 points).
(c) Implement bidirectional LSTMs for the encoder and report BLEU score for the test set. What is difference in BLUE score compared to the encoder with unidirectional LSTMs and explain why (10

points).
(d) (bonus) Implement *Additive* attention in (Bahdanau et al., 2014) instead of the basic dot-product attention. That is, for attention weight e_i for i-th encoder hidden state h_i and decoder hidden state s,

$$e_i = v^\intercal \tanh(\boldsymbol{W}_1 h_i + \boldsymbol{W}_2 s) \in \mathcal{R}$$

where $h_i \in \mathcal{R}^{d_1}$ and $s \in \mathcal{R}^{d_2}$, and $\boldsymbol{W}_1 \in \mathcal{R}^{d_3 \times d_1}$, $\boldsymbol{W}_2 \in \mathcal{R}^{d_3 \times d_2}$ are weight matrices and $v \in \mathcal{R}^{d_3}$ is a weight vector. Report BLUE score for the test set. What is difference in BLUE score compared to the basic dot-product attention and explain why (10 points).
(e) (bonus) Replace the embeddings with pre-trained word embeddings such as word2vec or GloVe and report BLUE score for the test set. What is difference in BLUE score compared to the previous model and explain why (10 points).

2 Implementing Transformer model (70 points)

(a) We provide jupyter notebook file (`assn3_2_transformer.ipynb`) for implementing the model. Implement encoder, decoder, and training and inference code as instructed in comments. Note that, for positional encoding, we use positional embedding as in BERT unlike the positional encoding in the original Transformer paper. Report your translation results for a few examples in the validation set and attention visualization of them. Report BLUE score for the test set (60 points)
(b) Tune your model and code to get the best performance you can get, and report BLUE score for the test set. Explain why your setting achieves this. Note that training of your model should be done in reasonable time at Google Colab (10 points).