Assignment 6: Machine Translation

11-411/11-611 NLP Teaching Staff

Due: March 23, 2023

1 Introduction

Machine Translation (MT) is one of the most important applications of Natural Language Processing. In this assignment, we will be learning how to train and evaluate our own machine translation system.

2 Learning Objectives

- Create a machine translation NLP Encoder-Decoder Model.
- Become familiar with creating models with PyTorch by stacking layers of functions.

3 Task 1: Programming (60 points)

Actual MT models require huge amounts of data and lots of computation to generate reasonable results. In this assignment, we will be building a toy model to train on a very small translation dataset. We recommend using Google Colab for this assignment. You can also try using your own GPU or CPU if you have access to computing resources. The notebook currently does not work if you use an Apple Silicon Mac with a GPU due to a PyTorch bug with LSTM back-propagation.

We provide you with preprocessed English-Mandarin parallel data split into three sets: train, valid, test. You should train the model on the training set, test it on the validation set while training, and generate and submit translations on the test set. You should upload `data.zip` to Colab and unzip it there.

Use the starter notebook provided to you. Make sure you are using the GPU by selecting Runtime $\rightarrow$ Change runtime type $\rightarrow$ Hardware accelerator $\rightarrow$ GPU.

In the starter notebook provided to you, fill in the code snippets under the sections

marked **# TO-DO**. Replace placeholders such as: `model = None` or `pass` with actual code. At the end of this assignment, you would have trained and evaluated a simple encoder-decoder model for machine translation.

You are given 2 files:

1. `encoder_decoder.ipynb` : Starter notebook to easily run and experiment with your code.
2. `encoder_decoder.py`: Python file that has the same code as the notebook. Please fill in the same code snippets as you did in the notebook and submit this file to Gradescope.

The notebook writes results into 6 files:

1. `en-zh.en.model`
2. `en-zh.zh.model`
3. `en_zh_encoder.pt`
4. `en_zh_decoder.pt`
5. `losses_en_zh.txt`
6. `preds.zh`

Please submit those files on Gradescope.

4 Task 2: Analysis (40 points)

Please submit your answers as a PDF on Gradescope and make sure to assign pages to answers.

4.1 Subtask 1 (10 points)

Use the default hyper-parameters, train a baseline MT model.

1. Plot the loss values on train and validation sets. What trends can you observe? What is this behavior called?
2. Report the BLEU and CHRF scores on the validation set.
3. Generate translations with the first 5000 lines of the training set. Report BLEU and CHRF scores. Can you explain the differences between the two sets of scores?

4.2 Subtask 2 (15 points)

Having trained the baseline model, you can now try to improve the performance by tuning the hyper-parameters.

1. List at least three hyper-parameters that you think might improve the model's performance.
2. Implement one of the changes. Report the final BLEU and CHRF scores.
3. Did the performance improve? List some possible reasons as to why or why not.

4.3 Subtask 3 (15 points)

In this assignment, we trained a simple encoder-decoder model. The following questions pertain to the model architecture decision.

1. We use an LSTM to model a sequential relationship. Why would you choose LSTM over vanilla RNN?
2. What part of the encoder does the decoder have access to?
3. Do you think this is "good enough"? If not, what other approaches would you use to model the encoder-decoder approach?

5 Deliverables

We will be using Gradescope for the submission of this homework. You will be submitting the following files to Gradescope

1. `analysis.pdf` – your answers for the Task 2 analysis. Please submit a pdf for this part of the assignment and assign pages to the question/answers to get credit.
2. `encoder_decoder.py` – python file containing the model and other details that are filled by you.
3. `en-zh.en.model` – trained Sentencepiece English model
4. `en-zh.zh.model` – trained Sentencepiece Mandarin model
5. `en_zh_encoder.pt` – trained encoder weights.
6. `en_zh_decoder.pt` – trained decoder weights.
7. `losses_en_zh.txt` – consists of the training and the validation losses for your final model. These are generated automatically from the script.
8. `test_preds.zh` – consists of Mandarin predictions from your model for each of the English sentences from the test set. You need to load `test.en` and run generations on the data.

6 Tips and Tricks

1. Please be wary of Google Colab, sometimes it may disconnect and please make sure that it does not disconnect during the training phase of the model. You can keep the session active by performing a click every few minutes. Getting a Mac app like Amphetamine works well if you select the option: Quick Preferences $\rightarrow$ Move cursor after 5m of inactivity.

2. Use your personal GPU and don't let your laptop sleep.

3. Have a look at PyTorch documentation for each layer and understand the input and output of each layer before plugging in the layers. Otherwise, you will see errors for layer size mismatch.