11-695: Competitive Engineerng
Assignment 2: (Fake) Machine Translation

Spring 2018

Abstract

In this assignment, you will implement a tiny neural machine translation system in TensorFlow [Abadi et al., 2015]. The starter code is provided for you, with all the data loading mechanisms implemented. Your job is to understand the starter code and implement a sequence-to-sequence model [Sutskever et al., 2014], and potentially with attention [Bahdanau et al., 2015], to translate English into Chinese. It is required that your implementation works on *batches of sentences,* as it is an important practice that significantly speeds up deep learning algorithms. However, the size of the dataset is small enough for you to achieve the requiredment even without any GPUs.

This assignment is due April 12th, at 11::59pm. Submission via email.

1 Code and Dataset

Install TensorFlow. If you have not already, please navigate to TensorFlow's site and follow their instructions to install the framework. The instructions are at

https://www.tensorflow.org/install/

Their instructions should be sufficient to install TensorFlow and all of its dependencies on your system. It is possible to complete this assignment without using any GPU, so you do not need to install CUDA or anything related to GPU programming. However, if you wish to, you are more than encouraged to install TensorFlow GPU, which will make your implementation much faster.

Compared to Assignment 1, the starter code of Assignment 2 uses `tf.dataset`, which is a recently added feature fo TensorFlow. Thus, please make sure that your version of TF is new enough.

Download the Dataset. You will be working with a subset of the IWSLT-2014 dataset [Cettolo et al., 2014]. We have preprocessed that data for you and have left out the sentences that are longer than 15 words. The cleaned version of the dataset can be downloaded at

http://www.cs.cmu.edu/afs/cs/user/hieup/www/11695/en_zh.zip

There are $64,221$ pairs of parallel sentences for training, and 253 pairs of sentences for evaluation. The vocabulary of each language consists of 5000 tokens, where the tokens do not necessarily to a word, but maybe a sub-word unit.

The starte code implements some data preprocessing procedures for you. In particular, we have written the code to read the dictionary files and to turn any pairs of sentences into tensor of integers, which you can use to look up for embeddings, e.g. by `tf.nn.embedding_loopup`.

Starter code. The starter code for your project is available at

http://www.cs.cmu.edu/afs/cs/user/hieup/www/11695/starter_code_2.zip

After downloading and unzipping the code and the data, you can navigate to your code directories. As with the last assignment, you will see the folders `src` and `scripts`. You will be writing code in the `src` and running your work using the shell-script files in `scripts`. Please make your code runnable simply by calling `./scripts/neural_mt.sh`, as this is what we will do to grade your work.

The starter code implements a working pipeline, but there is no baseline model implemented. In fact, the start code just returns 0 for all of your `logits`.

2 Your Work and Gradings

The entry point of the program is in the file `src/main.py`. From this file, relevant modules from `src/models.py`. It is your responsibility to read the code and figure out all the relevant points.

You are required to implement and train a sequence-to-sequence model on the given. The breakdown of the your work is as follows

1. Implement a sequence-to-sequence model that runs in *batches*. This means that your implementation has to be able to consider sentences of different lengths and "stich" them together into a batch to process them. This part is worth **20 points**. Your implementation should consist of the following components:

   - A mechanism to compute the loss and other training components.
   - A mechanism to *produce* translations. For this part, typically people use beamsearch [Sutskever et al., 2014]. However, we suggest that you use a simple greedy strategy that works reasonably well. Specifically, at each step in processing the sequence in your target language, you can just pick the output with the largest probability.

   We will test your work by running your program with two different batch sizes, and your program should produce the same results for both.

2. Train your model to achieve a reasonable performance on the given dataset. This part is worth **30 points**. We will evaluate your model's performance using two metrics:

   - Perplexity. This metric measures how *uncertain* the model is about predicting the *correct output*. Specifically, if your source sentence is $\mathbf{x} = \{x_1, x_2, ..., x_{|\mathbf{x}|}\}$ and your target sentence is $\mathbf{y} = \{y_1, y_2, ..., y_{|\mathbf{y}|}\}$, then perplexity is measured as

   $$\text{PPL}(\mathbf{x}, \mathbf{y}) = \left(\prod_{i=1}^{|\mathbf{y}|-1} p(y_{i+1} | y_{1 \cdots i}, \mathbf{x}) \right)^{|\mathbf{y}|-1} \tag{2.1}$$

   - BLEU score [Papineni et al., 2002]. Please print an output file of your translation, one per row. We will compare your output with the correct output.

   The required performance is that you reach **10.0** perplexity (lower is better) and **15.0** BLEU score (higher is better).

   These two metrics are very correlated. Lower perplexity leads to higher BLEU scores. Therefore, we will give you extra credits based on BLEU score only. Specifically, for each 0.25 BLEU score that you could improve above 15 BLEU, you will have **+1 point** for extra credit, rounding up.

Academic Integrity. As normal, you are encouraged to discuss with your friends and the instructors. Anything they tell you, you can use. However, looking at other's code should not happen at all cost.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Mauro Cettolo, Jan Niehues, Sebastian Stuker, Luisa Bentivogli, and Marcello Federico. Report on the iwslt 2014 evaluation campaign. Technical report, 2014.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. 2002.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.