11-695: AI Engineerng
Assignment 2: Machine Translation

Spring 2019

Abstract

In this assignment, you will implement a tiny neural machine translation system in TensorFlow [Abadi et al., 2015], in Eager Execution mode. The starter code is provided for you, with all the data loading mechanisms and training driver implemented. Your job is to understand the starter code and implement a sequence-to-sequence (Seq2Seq) model [Sutskever et al., 2014], and potentially with attention [Bahdanau et al., 2015], to translate French into English. It is required that your implementation works on *batches of sentences,* as it is an important practice that significantly speeds up deep learning algorithms.

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. Please use at least the version 1.13 for Eager Execution and other related features.

Datasets. There are 3 datasets provided to you along with the code file: Toy Reverse, English-Spanish and English-French.

1. Toy Reverse[1]: In practice, it's always a good idea to start with some toy dataset. This dataset contains 10K training pairs and 2K validation ones. Each single line contains a pair: source sentence and target sentence (which is the source being reversed). The Vocabulary size is only 24 and should be a good validation of your implementation. We recommend you play with this dataset first to make sure your model works well before moving on.

2. English-Spanish: This dataset has the similar format to Toy Reverse but for pairs of English and Spanish sentences.

3. English-French: This one is also similar but with French and is also the biggest one of the three. Along with English-Spanish, this dataset is extracted from: http://www.manythings.org/anki/

[1] https://google.github.io/seq2seq/

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

http://www.cs.cmu.edu/~htpham/11695-s19/assignment.html

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 `TODOs` and hints which suggest what you should or should not do. You will be mainly working one 2 files: `train.py` and `models.py` and no need to change the other ones.

Unlike the first assignment, this one is a little bit more challenging in the perspective that it does not provide you a dummy and driver where you could run the pipeline end-to-end without doing anything.

2 Your Work and Gradings

The entry point of the program is in the file `train.py`. From this file, relevant modules will be called. It is your responsibility to read the code and figure out all the relevant points. We also provided some `TODOs` and hints, however, to make your tasks less challenging.

You are required to implement and train a Seq2Seq model on top of the given code base. The breakdown of the your work is as follows

1. Implement a Seq2Seq model that runs in *batches*. We already did the padding and batching for you so you just focus more on the model itself and how to train it properly w.r.t what you have learned in class. Those are the breakdown details:

   - `models.py`: You are to implement the Encoder and Decoder part of the Seq2Seq, extended from `tf.keras.Layers`. We expect you do use the `tf.while_loop` and the mechanism at the low level. You are also given the `lstm` inference function for its use in the Encoder and Decoder. If you plan to use high-level APIs from keras for submission, you will get Zero bonus point, although it's a good idea to use it to test. For example, if you find it's easy, you can use it for Decoder and test your lower-level implementation for the Encoder, or vice versa.
   - `train.py`: You have to implement the Teacher-Forcing training mechanism in the `teacher_forcing` method. At test time, however, you have to use `evaluate` method for you have no ground truths and have to live with your predictions.
   - Evaluate your work by yourself: Periodically, the train driver saves your model and randomly presents 2 translations for you to observe the overview quality of the training. Furthermore, it also carries out the full validation translation and generate a file named `translated_n.txt` wher `n` is the final epoch number. It is very slow for this operation so we just set it up at the very end of the training, esp. for the English-French dataset. Nonetheless, it is not too long for the Toy Reverse dataset and you can test it at the end of a single epoch or however you want. When you have that translated file(s), you can calculate the BLEU score(s) as follow: `cat translated_10.txt | sacrebleu source.txt` in your dataset folder, using `sacreBLEU`[2]. The very first result of that script is what we use to grade you.

   You are expected to implement a single RNN for Encoder and another one for the Decoder. For Decoder, it is recommended that you can implement Attention model(s) Bahdanau et al. [2015], Luong et al. [2015] which will greatly affect the final results.

   Although not required, you can implement any kind of dropout you want. We recommend you leave it to the end if you still have time. Similarly for such extentions as stacked layers of LSTM, bidirectional layers or others.

[2] https://github.com/mjpost/sacreBLEU

2. Train your model to achieve a reasonable performance on the given datasets. You are expected to reports your performances on the Toy Reverse and English-French datasets. There are 2 metrics that we use to evaluate your work:

   - 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.

   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. These are the baselines:

   - Toy Reverse: With no attention, you should achieve at least BLEU **50.0** and with attention, it should be at least BLEU 90.0. A good implementation should show that the test BLEU can be 81.7 (no attention) and 99.9 (attention), respectively after 30 epochs.
   - English-French: The baseline is BLEU **10.0** (no attention). A good implementation should have at least BLEU 24.0 (no attention) with 30,000 training pairs.

3 Reports

As usual, you are required to submit your log files of training, your statistics on any method on any dataset that you work on. The limit is maximum 2 pages. We will give more credits if you have a good quality one.

4 Gradings

The grading breakdowns are as follows with totally 110 points:

1. Pass the baselines of Toy Reverse: **60 points**
2. Pass the baseline of English-French without attention: **20 points**
3. Exceed the baseline for English-French with attention: **10 points**
4. Report: **10 points**
5. Extra credits: up to **10 points** . As usual, we will not limit any creative ideas from you; feel free to apply any ideas that you see fit.

5 Submission

Submit the following to our shared instructor email at: cmu.11695.sp2019@gmail.com

- Your folder of code.
- Your real log of running each of your models. You can export log of running simply by using this syntax: `python3 train.py 2>&1 | tee log.txt`.

- Your report.

Note: Your code must be runnable directly with Python3 and Tensorflow. Any extra packages might be super useful but is prohibited. If you want to use any of new packages on the internet, check with the instructors first. Furthermore, if your code has a special instruction to run, please detail it. We will give you zero for each part where your code is not runnable. The TAs will be running each of your code so please be considerate to them as well.

6 Bonus

As circulated, we give extra credits for top 3 performers for this assignments (total best accuracies combined for 2 datasets). We also reserve our right to change this bonus scheme based on your real submissions.

As usual, we urge you to start early and acquire help from instructors soon.

7 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.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

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.

Revision

1. Mar 21: First release
2. Apr 10: Fixed some typo and edit some illogical scheme in grading