11-695: AI Engineering
Assignment 2: Neural Machine Translation

Spring 2020

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 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, Luong et al., 2015], to translate **French (target) into English (source)**. It is required that your implementation works on *batches of sentences* for it can significantly speed up deep learning algorithms and so is important in practice.

1 Code and Dataset

Install TensorFlow (of the required version at least 2.0). 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 encouraged to install TensorFlow GPU (See "Additional setup" section from the link above), which will make your implementation much faster, hence saving much of your time. Otherwise, another fast and convenient solution is to use some public deep learning AMI on AWS.

Datasets. There are 2 datasets provided to you along with the code file: Toy Reverse 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, thus this dataset should be a good validation mechanism of your implementation. Consequently, we recommend you play with this dataset first to make sure your model works well before moving on. Furthermore, a part of your grade is also evaluated on this dataset. Set FLAGS.is_toy to True and FLAGS.data_path to 'toy-reverse' to use this dataset.

2. English-French: This dataset has the similar format to Toy Reverse but for pairs of English and French sentences. . It is extracted from: http://www.manythings.org/anki/. Set FLAGS.is_toy to False and FLAGS.data_path to 'en-fr' to use this dataset.

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

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

http://aieng.cs.cmu.edu/assignment.html

After downloading and unzipping the code and the data, you can navigate to your code directories. Same as 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 others.

Unlike the first assignment, this one is a little bit more challenging in the perspective that it does not provide you a runnable dummy codebase which you can run the pipeline end-to-end without doing anything.

2 Your Jobs

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. But as said above, we already 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 required to implement the Encoder and Decoder part of the Seq2Seq, extended from `tf.keras.Layers` and `tf` library in general. There are **4 main suggestions** here:

     (a) Unidirectional RNN implementation for both Encoder and Decoder

     (b) Attention model: this one is the most tricky part so implement carefully. After that, everything is much simpler.

     (c) Bidirectiontal RNNs. You can use keras.Bidirectional wrapper. However if you create multilayer RNN and use the wrapper, forward and backward output from previous layer are combined according to your `merge_mode` argument and passed to next layer, which is different from pass forward/backward hidden state separately to forward/backward RNN at next layer

     (d) Stacked (multilayer) RNNs

     Note: Submit the model that has best performance is enough. You are not required to submit 4 models. However, it is recommended that you can implement Attention model(s) [Bahdanau et al., 2015, Luong et al., 2015] which will greatly affect final results. In our code base, we implemetned Bahdanau attention with modification that we use hidden state from `t` instead of `t-1` to calculate attention score. You can implement any type of want and modify our code structure accordingly. We recommend you to investigate different attention types and try them out as you may find inconsistencies in attention implementation on the internet.

   - `train.py`: There are two main jobs here. First, you are required 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 (and deal) with your predictions.

   - Evaluate your work by yourself: Periodically, the train driver saves your model and randomly samples and presents 2 translations for you to observe the overview quality of the training qualitatively. Furthermore, it also carries out a full test translation from `target.txt` and

generates a file named `translated_n.txt` where `n` is the final epoch number. This operation is usually very slow 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 hence you can possibly test it at the end of a single epoch or however you want (set FLAGS.translate_only to True to perform test translation). When you have the translated file(s), you can calculate the BLEU score(s) as follow: `cat translated_n.txt | sacrebleu source.txt` in your dataset folder, using `sacreBLEU`[2]. The very first result of that script along with the training perplexity scores (already implemented in the code) are what we use to grade you.

Important Note: We require you to implement Attention module manually, so if you directly use the predefined API `tf.keras.layers.Attention`, we only give you 5 points (out of 15 points) in the grading rubric for the Attention part. In either way, that API is a great tool to validate your model.

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, your target sentence is $\mathbf{y} = \{y_1, y_2, ..., y_{|\mathbf{y}|}\} \sim p(\mathbf{y})$, then perplexity is measured as

   $$\text{PPL}(\mathbf{y}) = \left(\prod_{i=1}^{|\mathbf{y}|} p(y_i | y_{<i}) \right)^{-\frac{1}{|\mathbf{y}|}} = 2^{\mathbf{H}(p(\mathbf{y}))} \tag{2.1}$$

   - BLEU score [Papineni et al., 2002] as instructed above using `sacreBLEU`. Also, please print an output file of your translation, one per row. We will compare your output with the correct output.

3. When you run your program, there is a flag, which is `is_toy` for you to switch between the toy dataset and the other. Also there are many flags in the default driver that you can change *w.r.t* your needs, so please make use of them accordingly before you run the first program with toy dataset. Last, you can use the two bash scripts that we prepared, namely `train_toy.sh` and `train_fr.sh` that have all flags properly set.

3 Baselines

You are required to pass the following 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 (with attention), respectively after about 20-30 epochs.
- **English-French**: The baseline is BLEU **10.0** (without attention) and **20.0** (with attention). A good implementation should have at least BLEU 26.0 (with attention) after about 10 epochs.

As usual, we urge you to start early and acquire help from the course staffs soon.

4 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 high quality one.

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

5 Gradings

The grading breakdowns are as follows with totally 100 points with 10 potential credit points:

1. Single-layer RNN without attention: **20 points**
2. Attention: **15 points**
3. Bidirectional RNN: **5 points**
4. Stacked (multi-layer) RNN: **5 points**
5. Pass the baselines of Toy Reverse (depending on whether you have attention): **20 points**
6. Pass the baselines of English-French (depending on whether you have attention): **20 points**
7. Report: **15 points**

1 is considered passed if you already have an attention model. Submit the model that has best performance and we will grade you accordingly.

6 Submission

All submissions must be made to Canvas individually, including:

- 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 (of version at least 2.0). Any extra packages might be super useful but is prohibited, and should not be needed. But if you insist 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.

7 Academic Integrity.

As usual, you are encouraged to discuss with your friends and the instructors. Anything they tell you, you can use. However, looking at other people's codes 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 01: First release.
2. Mar 18: Fix some typos, grading explanation and perplexity.
3. Mar 22: Fix some unclear descriptions.