Machine Translation Marathon 2016 – Lab Session – Neural Machine Translation

Rico Sennrich <rico.sennrich@ed.ac.uk> Tom Kocmi<kocmitom@gmail.com>

15. September 2016

1 Neural Machine Translation with Nematus

Nematus[1] is a theano-based toolkit for training and decoding with attentional encoder-decoder networks [Bahdanau et al., 2015]. It is a fork of the dl4mt-tutorial[2], and was used by the University of Edinburgh for their top-ranked WMT16 submissions [Sennrich et al., 2016a]. Pre-trained models are available for the language pairs EN↔{CS,DE,RO,RU}.[3]

In this lab session, you will learn how to do the following:

1. install Nematus
2. translate a document with Nematus and an existing NMT model
3. adapt an existing NMT model to a new domain via continued training [Luong and Manning, 2015, Sennrich et al., 2016b]

1.1 Install Nematus

First, choose where you want to install Nematus. It is highly recommended to install Nematus on a machine with a CUDA-capable NVIDIA GPU (with at least 6GB memory). You will not need administrative privileges to install Nematus itself, but administrative privileges are required to install some prerequisites, such as nvidia-docker, or CUDA and CUDNN. If you do not have access to a machine where all dependencies are installed, or where you have administrative privileges, you can use an Amazon EC2 machine.

If available, we recommend installation via Docker, but we provide installation instructions with and without Docker.

1.1.1 installation in Docker container

1. install docker and nvidia-docker: `https://github.com/NVIDIA/nvidia-docker`. On Amazon EC2, follow these instructions to create a GPU instance and install nvidia-docker: `https://github.com/NVIDIA/nvidia-docker/wiki/Deploy-on-Amazon-EC2`.
2. download Nematus:

   ```
   git clone https://github.com/rsennrich/nematus
   ```

3. build docker image:

   ```
   cd nematus && docker build -t nematus-docker -f Dockerfile.gpu . && cd ..
   ```

[1] `https://github.com/rsennrich/nematus`
[2] `https://github.com/nyu-dl/dl4mt-tutorial`
[3] `http://statmt.org/rsennrich/wmt16_systems/`

1.1.2 installation without Docker (in Python virtual environment)

1. install CUDA (and ideally CUDNN)
2. install theano in python virtual environment

   ```
   pip install --user virtualenv #install virtualenv
   virtualenv virtual_environment #create environemnt
   source virtual_environment/bin/activate #start environment
   pip install numpy numexpr cython tables theano ipdb #install theano
   ```

3. you may need to install BLAS library and other dependencies
   on Debian Linux:

   ```
   sudo apt-get install liblapack-dev libblas-dev gfortran libhdf5-serial-dev
   ```

4. install Nematus and scripts for preprocessing

   - mosesdecoder (just for preprocessing, no installation required)

     ```
     git clone https://github.com/moses-smt/mosesdecoder
     ```

   - subword-nmt (for BPE segmentation)

     ```
     git clone https://github.com/rsennrich/subword-nmt
     ```

   - Nematus (DL4MT fork; for training NMT)

     ```
     git clone https://www.github.com/rsennrich/nematus
     cd nematus && python setup.py install && cd ..
     ```

2 Translating with Nematus

Pre-trained models are available at `http://data.statmt.org/rsennrich/wmt16_systems/` for 8 translation directions. This tutorial assumes that we work with the translation direction English→German.

1. download model and enter directory:

   ```
   wget -r --cut-dirs=2 -e robots=off -nH -np -R index.html* \
   http://data.statmt.org/rsennrich/wmt16\_systems/en-de/
   cd en-de
   chmod a+x translate.sh
   ```

2. if you installed Nematus via Docker, start the Docker instance:

   ```
   nvidia-docker run -v ‘pwd‘:/playground -it nematus-docker
   ```

3. if you did not install Nematus via Docker, adjust paths in *translate.sh* to point to your location of nematus, mosesdecoder and subword-nmt

4. translate a test sentence:

   ```
   echo "This is a test." | ./translate.sh
   ```

3 Adapting an Existing Model

The pre-trained models are trained on data provided by the WMT shared translation task[4], and optimized for News texts. In this section, you will adapt the model to the TED data set, which is the test domain of the IWSLT shared translation task[5]. We will use the pre-trained model from the previous section.

1. download training and development data:

   ```
   wget https://wit3.fbk.eu/archive/2016-01//texts/en/de/en-de.tgz
   mkdir iwslt
   tar -xf en-de.tgz -C iwslt
   ```

2. preprocess training data. While you are free to define your own preprocessing when training a new model, we highly recommend to use the preprocessing that was used for the original model when continuing training thereof. For this case, execute the following commands for preprocessing (as seen in `http://data.statmt.org/rsennrich/wmt16_systems/en-de/translate.sh`):

```
#English (source side)
cat iwslt/en-de/train.tags.de-en.en | \
/path/to/mosesdecoder/scripts/tokenizer/normalize-punctuation.perl -l en | \
/path/to/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en -penn | \
/path/to/mosesdecoder/scripts/recaser/truecase.perl -model truecase-model.en | \
/path/to/subword-nmt/apply_bpe.py -c ende.bpe > corpus.bpe.en

#German (target side)
wget http://data.statmt.org/rsennrich/wmt16_systems/de-en/truecase-model.de
cat iwslt/en-detrain.tags.de-en.de | \
/path/to/mosesdecoder/scripts/tokenizer/normalize-punctuation.perl -l de | \
/path/to/mosesdecoder/scripts/tokenizer/tokenizer.perl -l de | \
/path/to/mosesdecoder/scripts/recaser/truecase.perl -model truecase-model.de | \
/path/to/subword-nmt/apply_bpe.py -c ende.bpe > corpus.bpe.de
```

3. determine your training configuration. You can use the file at `https://raw.githubusercontent.com/rsennrich/wmt16-scripts/master/sample/config.py` as a starting point. Some parameters, like the vocabulary size, must fit that of the original model – you can check the model parameters of the original model in the corresponding JSON file [6].

   - set *datasets* to the preprocessed training sets from the previous step.
   - ensure that the *dictionaries*, vocabulary size (*n_words* and *n_words_src*), and size of the embedding/hidden layers (*dim* and *dim_words*) correspond to that of the original model.
   - to continue training an existing model, *saveto* must point to the original model, and *reload_* must be True.
   - add an option "max_epochs=1" to only perform training for one epoch on the IWSLT dataset. The model would quickly overfit with multiple epochs, and one epoch should take 30-60min, depending on your hardware.
   - remove *external_validation_script* for now[7], and set *valid_datasets* to None (validation is generally useful, but we skip it in this tutorial for simplicity).
   - the model should train well on a GPU with 8GB of memory; to run on a 6GB GPU, there are various ways to reduce memory consumption, for instance reducing the *batch_size* to 60.

[4] `http://www.statmt.org/wmt16/`

[5] `https://sites.google.com/site/iwsltevaluation2016/mt-track`

[6] `http://data.statmt.org/rsennrich/wmt16_systems/en-de/model.npz.json`

[7] This is used for BLEU early stopping; look at `https://raw.githubusercontent.com/rsennrich/wmt16-scripts/master/sample/validate.sh` for an example.

4. start training (this will overwrite your original *model.npz*, so you should make a copy first):

```
THEANO_FLAGS=mode=FAST_RUN,floatX=float32,device=gpu,lib.cnmem=0 python config.py
```

5. you can translate a validation set from IWSLT before and after adaptation to observe the difference in BLEU. IWSLT uses the NIST scoring tools (mteval-vX.pl) for scoring BLEU, and input and references are wrapped in XML. Here's an example showing how to translate and score a test set, assuming that your translation system is *./translate.sh*

```
/path/to/mosesdecoder/scripts/ems/support/input-from-sgm.perl < iwslt/en-de/IWSLT16.TED.tst2013.en-de.en.xml > tst2013.en
./translate.sh < tst2013.en > tst2013.out
/path/to/mosesdecoder/scripts/ems/support/wrap-xml.perl german \
iwslt/en-de/IWSLT16.TED.tst2013.en-de.en.xml < tst2013.out > tst2013.out.xml
/path/to/mosesdecoder/scripts/generic/mteval-v13a.pl -c -s iwslt/en-de/IWSLT16.TED.tst2013.en-de.en.xml \
-r iwslt/en-de/IWSLT16.TED.tst2013.en-de.de.xml -t tst2013.out.xml
```

On *IWSLT16.TED.tst2013*, we obtained a BLEU score of 0.2677 with the pre-trained model – you should be able to reproduce this number exactly. After one epoch of continued training, we obtained a BLEU score of 0.3007 – your score will vary because the datasets are randomly shuffled before training, which makes results non-deterministic.

References

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[Luong and Manning, 2015] Luong, M.-T. and Manning, C. D. (2015). Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation 2015*, Da Nang, Vietnam.

[Sennrich et al., 2016a] Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 368–373, Berlin, Germany. Association for Computational Linguistics.

[Sennrich et al., 2016b] Sennrich, R., Haddow, B., and Birch, A. (2016b). Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.