Machine Translation: Practical Work 4

Sharid Loáiciga

In this assignment, you will train Neural Machine Translation (NMT) models using BPE and beam search. **Caution!** Note that this is a more invested assignment, with models that take longer to train. You have three weeks to complete it. If you run into problems one day before the deadline, you might not have enough time to recover, so please start early!

1 Set up

You will be working in the CogSys GPU server. To log in, you need the `ssh` command:

```
ssh USER@jarvis.ling.uni-potsdam.de
password:  PASSWORD
ssh pm-nmt1 or pm-nmt2
```

To end your session at the end of your work use `exit`.

2 Choose a language pair

For this assignment, you will train NMT models for a language pair of your choice. A selection of languages from the OpenSubtitles collection from the OPUS corpus (`http://opus.nlpl.eu/`) has been made available in the shared folder ˜/`data/shared_folder/OpenSub/`. Select a language pair in a way that you are able to evaluate the target language output. You can also combine several corpora for the same language pair, provided that they are of similar genres.

Before starting the experiments, we need to split the corpus into a training set, a validation set and a test set. The validation and test sets should contain about 5,000 lines each. The training set should have no less than 0.5M lines. Shorten the training set to 500,000 lines if necessary. Use the `head` and `tail` tools to split the corpus, and **make sure that the three sets do not overlap**.

In the report, specify what corpus and language pair you chose and how you performed the data splitting.

3 Preprocessing

Tokenize all files using the script introduced in the last assignment. Also create a truecasing model for each language and truecase al files.

You will train three different NMT models with different subword splitting schemes:

- A model without any subword splitting
- A model with 3000 BPE merge operations
- A model with 32000 BPE merge operations

For the second and third model, you need to add another preprocessing step to split the words into subword units. As for truecasing, you first have to train a model (on the training data only) and then apply it to the train, dev and test data.

To train a BPE model[1], use the following command:

```
subword-nmt learn-bpe -s 3000 < training.es.true.txt > bpe3k.es.codes
```

To apply the BPE model to a data file, use the following command:

```
subword-nmt apply-bpe -c bpe3k.es.codes < training.es.true.txt > training.es.bpe3k.txt
```

Repeat these commands so that you get 4 models (2 per language and 2 per number of merge operations) and 12 data files (3 files x 2 languages x 2 models).

4 Data conversion and training

For each of the three models, you now have to convert the corresponding data and start training. For the former, use the `onmt_preprocess` command from the OpenNMT library, as instructed in the previous assignment. For the latter, use the `onmt_train` command. This is a suggestion for the parameter settings:

```
onmt_train -data data-lang1-lang2 -save_model model-lang1-lang2 -train_steps 100000
-save_checkpoint_steps 5000 -valid_steps 5000 -global_attention none -input_feed 0
-dropout 0.0 -world_size 1 -gpu_ranks 0 -layers 2 -rnn_size 300 -word_vec_size 300
```

These models can take some **12 hours** to run and use about 8 GB of memory. You may use the `nohup` and `&` commands to let your models train while you are off the terminal. For example:

```
nohup onmt_train -data data-lang1-lang2 -save_model model-lang1-lang2 -train_steps
100000 -save_checkpoint_steps 5000 -valid_steps 5000 -global_attention none \
-input_feed 0 -dropout 0.0 -world_size 1 -gpu_ranks 0 -layers 2 -rnn_size 300 \
-word_vec_size 300 &
```

5 Testing

Test the three models by translating the respective test sets. You can use the same `onmt_translate` command as in the last assignment, using the last model file created during training. (It doesn't matter if the training stops before having reached the 100,000 training steps.)

Note that the BPE splitting codes must be undone before detokenizing. This is done with the `detokenizer.perl` script within `/data/shared_folder/preprocessing-tools`. This is a suggestion of command:

```
sed -r ‘s/(@@ )|(@@ ?$)//g’ < pred.bpe.lang2 | detokenizer.perl -u -l \
lang2 > pred.detok.lang2
```

Look at the three output files and discuss the following questions in the report: Which one looks best? Which one looks worst? Why?

Compute the BLEU scores for the three output files and list them in the report. Do the scores support your intuitions?

6 Beam search

Pick the best of the three trained models for this exercise. You will look at the impact of different beam sizes.

[1]The code and documentation of the `subword-nmt` tool can be found here: `https://github.com/rsennrich/subword-nmt`

In exercise 4, you have produced translations with the default beam size of 5. Now, translate the test set again using a beam size of 10. To this end, add the parameter `-beam_size 10` to the `onmt_translate` command. Also translate the test set with a beam size of 1.

Compute BLEU scores for all three translations. How much do the translations differ? Can you spot particularities in the output of the files? Write your observations and results in the report.

Rerun the translation command `onmt_translate` with the following added options:

```
-beam_size 5 -n_best 5 -verbose
```

This will generate in your output file (*.out) the best 5 hypotheses of the beam search. Look at them and discuss your findings in the report.

7 Submission

Submit a PDF file in Moodle by Wednesday July 8th, 11:00pm. You should document your progress in the assignment. Answer the given questions but also note your observations and impressions.