ATMT Assignment 1

Training and evaluating a simple NMT system

Deadline: October 8, 14:00

Important submission info:

- **Submission format: pdf**
- Please name your submission files like this: olatusername_atmt_assignmentxx.pdf, e.g. `mmuster_atmt_assignment01.pdf`
- For this assignment, you are asked to submit your answers in a PDF file. Please do not hand in any other files.
- You need to submit the assignment on OLAT using the tab for assignment 1. Please hand it in on time. The submission deadline is given above. After this time, the tab will be closed.
- You are encouraged to work in groups of two. Please make sure every file clearly states **both** names. For pairs, only one person should submit the assignment.
- Note: this first assignment will not be individually marked, but will serve as a preparation for future assignments.

Your Task As a first exercise, you will train and evaluate your own NMT system. For this, we provide you with a toolkit to train your model as well as preprocessed data (normalized, tokenized and truecased). Even though the NMT system is very simple compared to full-fledged models and the provided data set is very small, the system will still take a considerable amount of time to train. We recommend that you train your model overnight on the CPU of your own machine and start early with this assignment.

1 Setting Up Your Environment

As a first step, clone, download or, better yet, 'fork' the repository from toolkit. As a second step, we recommend setting up a python virtual environment for working with the toolkit. We provide instructions on how to do this (with both `conda` and `virtualenv`) in the README of the toolkit's repo. We will use the toolkit regularly throughout the course, but you only need to follow the setup instructions once. Please note, in the README, you will also find example calls for the relevant scripts that you can adapt for your use. We also provide them here as an additional guide.

2 Training an NMT System

2.1 Data

To train an NMT system, we need training data. For this assignment we are using the Swedish-English parallel "infopankki" corpus, which is built from the InfoFinland website. We use a training set with 10,000 parallel sentences, a development and test set with 500 sentences each and a tiny training set for additional experiments with only 1,000 sentences. This can be considered a 'low-resource' setting and, in reality, much more data is used to train state-of-the-art NMT systems. We deliberately chose small data sets so that your models will train fast enough on a CPU.

The data that is already preprocessed and stored in a format that is readable for the toolkit. The training material can be found in the `data/en-sv/infopankki/` folder in the toolkit.

- `data/en-sv/infopankki/raw` contains the original data downloaded from OPUS that has been split into train, test and validation sets.
- `data/en-sv/infopankki/preprocessed` contains the normalized, tokenized and truecased version the data.
- `data/en-sv/infopankki/prepared` contains binarized, pickled files of the same data, which is expected by the toolkit. This folder also contains the dictionaries with the 4000 most frequent words in each language. This vocabulary size is deliberately small to speed up the training. In real-life scenarios, we would choose larger vocabulary sizes. We use these dictionaries to encode each sentences as a list of vocabulary ids that correspond to the tokens in the sentence. Tokens that are not in the dictionary are replaced with an "unknown" placeholder token.

2.2 Training

To train our models, we simply need to call `train.py` script and pass the relevant paths as command line arguments, e.g.:

```
python train.py \
    --data data/en-sv/infopankki/prepared \
    --source-lang sv \
    --target-lang en \
    --save-dir assignments/01/baseline/checkpoints
```

Depending on your operating system, you may need to adjust the paths passed to `--data` and `--save-dir`. The best and last checkpoints are saved to the save directory. These checkpoints are used later to load the best model and translate a text for evaluation. The training can take quite a long time because we are not training the model on a GPU. To avoid late submissions, please start early with this assignment. We recommend that you **train the model overnight** so you can finish the rest of the assignment the next day.

Note: we strongly recommend to first do a test-run on the `tiny_train` data set to ensure that everything is installed correctly and that training will run smoothly. To do this, simply set the flag `--train-on-tiny` when calling the `train.py` script. Since the `tiny_train.*` data set contains only 1,000 parallel sentences, model training should converge within about 15 minutes.

Submission: In your PDF report, please include the last lines of the training output log which show the epoch number, the training and validation loss and the validation perplexity before the training stopped.

3 Evaluating the NMT System

3.1 In-domain Evaluation

Once you have trained your NMT system, we want to evaluate how well it performs. For this, we need to translate our test set from Swedish to English. To do this, use the `translate.py` script, e.g.

```
python translate.py \
    --data data/en-sv/infopankki/prepared \
    --dicts data/en-sv/infopankki/prepared \
    --checkpoint-path assignments/01/baseline/checkpoints/checkpoint_last.pt \
    --output assignments/01/baseline/infopankki_translations.txt
```

The resulting translations will be saved in a file provided to the command line argument `--output`.

Next, we use an automatic measure, BLEU (bilingual evaluation understudy), to evaluate the performance of our system. To compute the BLEU score we use sacreBLEU, an evaluation tool that ensures that BLEU scores from different translation systems with different preprocessing are comparable. The inputs to sacreBLEU must be the raw text. This means that before

computing the BLEU score, we need to reverse any preprocessing steps we did before translation (tokenization and truecasing).

To postprocess the generated translations and produce raw text, use the `postprocess.sh` located in the `scripts` directory, e.g.:

```
bash scripts/postprocess.sh \
    assignments/01/baseline/infopankki_translations.txt \
    assignments/01/baseline/infopankki_translations.p.txt en
```

This will create a new file (e.g. `assignments/01/baseline/infopankki_translations.p.txt`) containing the raw translations, which can be used as the input to sacreBLEU for evaluation.

To evaluate our NMT system, execute the following command:

```
cat \
    assignments/01/baseline/infopankki_translations.p.txt \
    | sacrebleu data/en-sv/infopankki/raw/test.en
```

This computes the BLEU score of our translations given the reference translations from the `raw` data folder. The output is a dictionary-like object containing the BLEU score (`'score'`) as well as the n-gram precision values and the brevity penalty (BP) that are used to compute it (`'verbose_score'`).

3.2 Out-of-domain Evaluation

The translations we just produced are from an 'in-domain' test set, which means that the test set comes from the same distribution (the same parallel corpus) as the training material. In practice, this is not always guaranteed and can cause problems if the two domains are very different. To see the effect of this problem, we will now use our system trained on text from the InfoFinland website to translate a test set from another domain.

For this, you can choose whether to use parallel test sets from the Bible translations (`data/en-sv/bible/prepared`) or TED Talks (`data/en-sv/TED2020/prepared`). To translate these test sets with your model, use the same script as before, specifying the `--data` path to that of the desired out-of-domain test set, e.g.:

```
python translate.py \
    --data data/en-sv/bible_uedin/prepared \
    --dicts data/en-sv/infopankki/prepared/ \
    --checkpoint-path assignments/01/baseline/checkpoints/checkpoint_last.pt \
    --output assignments/01/baseline/bible_translations.txt
```

To compute the final BLEU scores, the postprocessing and evaluation steps are identical to before. However, make sure you change the path to the correct reference translations, e.g. `data/en-sv/bible/raw/test.en`.

Submission: Answer the following points in your PDF report with a short paragraph each:

- Create a table with the BLEU scores and the n-gram precision values that you computed on your indomain and out-of-domain translations.
- The BLEU score on the in-domain test set will be relatively high considering the model was trained on very little data. Take a look at the raw data sets. Which characteristics of the in-domain data could be responsible for a high BLEU score?
- Compare the model's performance on the in-domain test set vs. the out-of-domain test set. Why is the out-of-domain test set so much harder to translate? Support your answer with examples from the test set.
- Choose a language other than English that you know well. Find 3 words that may be translated differently into English depending on the context and provide examples. How do your examples fit into the discussion of in-domain vs. out-of-domain? Can you think of a possible way to ensure a specific translation for a word is used by an NMT model?

Please let us know in the OLAT forum if you have problems or something is unclear.
Good luck and have fun!