ATMT Assignment 1

Training and evaluating an NMT system on the ScienceCluster

Deadline: October 7, 14:00

Important submission info:

- **Submission format: zip**
- Please name your submission files like this: `<olatusername>_atmt_assignment<XX>.zip`, e.g. `mmuster_atmt_assignment01.zip`
- For this assignment, you are asked to submit your answers and files in a **zip** 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.
- This assignment should be completed **individually**.
- Note: this first assignment will **not be graded**, but completing it is **required** for the following assignments.

Your Task: As a first exercise, you will train and evaluate your own NMT system. As part of a pilot project, we have the opportunity to use UZH's ScienceCluster - a high-performance computing (HPC) environment usually reserved for research purposes, where we have access to powerful GPUs to train our own models.

For this, we provide you with a toolkit to train your model, the raw data required, as well as access to the cluster. As we do not have an unlimited number of GPU-hours, the models we train as part of the assignments won't rival the size and accuracy of state-of-the-art models, but will nevertheless achieve substantial and interesting results.

1 Connecting to the ScienceCluster

To connect to the ScienceCluster, you'll need to be connected to the UZH-network – either directly or via the university's VPN.

As a first step, connect to the ScienceCluster using the following command:

```
ssh -l shortname cluster.s3it.uzh.ch
```

where `shortname` is your UZH-shortname[1]. You will then be prompted to enter a password - this is your Active Directory password[2]. After that, you should be successfully connected to the cluster! For security reasons, you'll be disconnected after a certain period of inactivity.

1.1 Prerequisites

As a first step, fork the toolkit's Github repository. Then, on the cluster, change to your `data` directory and clone your fork:

```
cd data
git clone https://github.com/chamisshe/atmt_2025 #example - use your own forked repo!
```

As usual with a provided toolkit, it will have some dependencies that need to be installed. The ScienceCluster uses mamba (a reimplementation of conda) as its package manager. Follow the steps below (taken from the ScienceCluster docs) to install the required packages:

```
# load the gpu module
module load gpu

# request a 1-hour-long interactive session, which allows the package installer to
     see the GPU hardware
srun --pty -n 1 -c 2 --time=01:00:00 --gpus=1 --mem=8G --partition teaching bash -
    l

# (optional) confirm the gpu is available. The output should show basic
    information about at least
# one GPU.
```

[1] Not to be confused with your OLAT username - these are not the same for some reason. Unsure about your shortname? You can find it by logging into https://identity.uzh.ch/, under the tab *Technical Information*

[2] The same password that you use to log into your UZH-mail.

```
nvidia-smi

# use mamba (drop-in replacement for conda)
module load mamba

# create a virtual environment named 'atmt' and install packages
mamba create -n atmt -c pytorch -c nvidia --file requirements.txt

# activate the virtual environment
source activate atmt

# confirm that the GPU is correctly detected
python -c 'import torch as t; print("is available: ", t.cuda.is_available());
    print("device count: ", t.cuda.device_count()); print("current device: ", t.
    cuda.current_device()); print("cuda device: ", t.cuda.device(0)); print("cuda
    device name: ", t.cuda.get_device_name(0)); print("cuda version: ", t.version.
    cuda)'

# when finished with your test, exit the interactive cluster session
conda deactivate
exit
```

After this, you can activate your environment from anywhere, the only caveat being that `mamba` has been loaded before:

```
module load mamba
source activate atmt # (note that it is 'source' and not 'mamba'/'conda'!)
```

1.2 Data

To train an NMT system, we need training data. For this assignment, we are using the Czech-English parallel "CzEng 2.0" corpus[1]. We use a training set with 10 million parallel sentences, a development and test set with 5000 sentences each.

The data is stored under `/shares/atomt.pilot.s3it.uzh/`, which is one of the various file-systems on the ScienceCluster – you can learn more about the different types of storage here. To be able to access `/shares` like a regular directory, you must first create a symbolic link as follows:

```
ln -s /shares/atomt.pilot.s3it.uzh ~/shares
```

Notes:

- `atomt.pilot.s3it.uzh` is simply the project specific subfolder we all have access to – by default, everyone with access to it can read, write, and execute its contents.
- The resulting location of `/shares` is in the `/home` directory.

2 Training an NMT System

If we've followed the above steps correctly, we should be ready to train our first model. Let's start small with the toy-example, just to get familiar with the ScienceCluster. To do so, navigate to the directory where you cloned the toolkit to and run the `toy_example`:

```
sbatch toy_example.sh
```

This will submit a batch-job to slurm. To check the status of your submitted job(s), you can simply run the command `sacct`, which will give you a jobs id, run-time, state, and other more or less interesting information. But you're not seeing any output at all. Looking at the shell script, take note of the following line:

```
#SBATCH --output=toy_example.out
```

This will redirect the output of a submitted batch job to a specified file, which you can inspect using Unix' `cat` command.

Upon examining the output file, you may notice that although it isn't slow, it still takes a noticeable amount to train on a tiny amount of data. That is because we're not utilizing the GPUs for training yet.

Task 1 Examine the shell script, as well as the python scripts it calls for the preprocessing, training and translation steps. Familiarize yourself with the various arguments each script is called with, and figure out how to have it train on the GPU. The toolkit's README may prove helpful.

Note: Do not modify the lines starting with `#SBATCH`, as they are important information for Slurm, the cluster management that is used in the background.

Submission:
In a PDF, report the following:

- What were the steps you took to make GPU training work? (a concise list of steps is enough)
- Time per training epoch (excl. validation) on CPU vs. on GPU

2.1 Training a model

We should now finally be ready to train a real model. `assignment1.sh` follows the same outline as the toy-example, and it should be ready to go.
Run it.

```
sbatch assignment1.sh
```

Again, the same commands as in the toy-example apply to check a job's status and its output (though the actual filename is different). You are encouraged to examine the output file, see how training and validation performance changes from epoch to epoch, and how training, validation and evaluation are handled in the training script (`train.py`). Also, take a look at the final

translation. You can find out where it is by examining the `--output` argument passed to the translation script.

Submission:

- Include the output file from this training run in your submitted zip-folder
- Create a graph (e.g. using matplotlib or seaborn) to plot the training perplexity, validation perplexity, validation BLEU score and the final BLEU score. Include it in your PDF-report.

3 Some Experiments using the NMT System

As you may have noticed, the shell script for this assignment includes a section where the test set is translated. Of course, you can now use the trained model to translate different corpora. For the following translation experiments, choose **one** of the following two options:

1. Slovak is a language that is very similar to Czech. Find a Slovak-English sentence-aligned parallel corpus[3], and trim the file to 5000 sentence-pairs.
2. By looking at the website and paper of the CzEng 2.0 dataset, find out which other datasets it sourced its data from. Select two of the mentioned sources, find and download them[4], and trim each of them to 2500 sentence-pairs.

Note: Make sure the sentences are still aligned after trimming!

Next, create a new shell script that only does translation using the trained model. This time, use the data you just prepared as the source and reference, in order to calculate the model's BLEU score. Make sure you still use the GPUs! Refer to the ScienceCluster's documentation to potentially adjust the `sbatch`-parameters (1 GPU will be enough, but it may be worth it to scale your demands for time as needed).

Submission: Report the BLEU score that the model achieved on your dataset. How do you interpret the results? Take a glance at the translation output: Is there anything that stands out to you compared to the output generated on the original Czech test set?

4 Final Remarks

Since a lot has been changed from last year's exercises, there may be some starting difficulties. Don't hesitate to reach out on the forum, and we'll do our best to fix it!

References

[1] Tom Kocmi, Martin Popel, and Ondrej Bojar. "Announcing CzEng 2.0 Parallel Corpus with over 2 Gigawords". In: *arXiv preprint arXiv:2007.03006* (2020).

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

[3] Opus is a brilliant resource for this.
[4] Opus is a brilliant resource for this, too.