

Software Engineering for CSAI

Task 2 - AI environment, data and model

Project Group 1:

Dimitar Angelov

Quinten Cabo

Daan Franken

Stefano Scola

Gijs Thissen

Samuel van Wijhe

Software

Environment

Setting up the environment proved to be difficult. This was because a new version of OpenNMT has been released and therefore quickstart changed. The problems began when it was required to run the OpenNMT commands in the terminal. The first problem that was encountered was Torch refusing to install since it could not find version 1.6.0. Torch was able to find version 0.2.post. However, since the global environment was used and not the anaconda environment the program wouldn't run. Anaconda was installed instead and an environment was created called: nmt. To the aforementioned environment CUDA, PyTorch, and OpenNMT-py were added. These were added to PyCharm.

Data

Dataset

The dataset used for this project was the EuroParl dataset¹ that was described in (Tiedemann, J., 2012). (Tiedemann, J., n.d.) describes it as a parallel corpus extracted from the European Parliament web site by Philipp Koehn (University of Edinburgh). The main intended use is to aid statistical machine translation research.

The corpus consists of about 2M sentence alignments and of 10345 aligned documents.

The Moses file extension has been downloaded and extracted in the raw_data folder. It contained one file for the Dutch language and one file for the English language.

The maximum length of all the sentences contained in the English one is 3951, while for Dutch it was 3890. Furthermore, the minimum length of all sentences is 2 regarding both language files.

When calculating the length, white-space characters have been considered.

The sentences seemed to be aligned.







1. <http://opus.nlpl.eu/Europarl-v8.php>

Machine Translation Model

The script provided in the assignment (Shterionov, D., 2020) was used (train_test_dev.py). Please see Appendix A for more of the code used. This program was run in order to shuffle and split the data into train, test and validation. Apart from that it was also used to tokenize the corpus. This process took 20 minutes to run on a Geforce GTX 980.

```
onmt_preprocess -train_src train.src -train_tgt train.trg -valid_src dev.src -valid_tgt dev.trg  
-save_data readytotrain
```

The command shown above was run on the files shown below.

Name	Date modified	Type	Size
 dev.src	02-Oct-20 4:04 PM	SRC File	154 KB
 dev.trg	02-Oct-20 4:04 PM	TRG File	166 KB
 test.src	02-Oct-20 4:04 PM	SRC File	146 KB
 test.trg	02-Oct-20 4:04 PM	TRG File	160 KB
 train.src	02-Oct-20 4:04 PM	SRC File	145,972 KB
 train.trg	02-Oct-20 4:04 PM	TRG File	158,673 KB

```
(nmt) C:\Users\Gebruiker\se-csai-translation\EN-NL>cd data  
(nmt) C:\Users\Gebruiker\se-csai-translation\EN-NL\data>onmt_preprocess -train_src train.src -train_tgt train.trg -valid_src dev.src -valid_tgt dev.trg -save_data readytotrain  
[2020-10-02 17:48:33,405 INFO] Extracting features...  
[2020-10-02 17:48:33,406 INFO] * number of source features: 0.  
[2020-10-02 17:48:33,406 INFO] * number of target features: 0.  
[2020-10-02 17:48:33,406 INFO] Building 'fields' object...  
[2020-10-02 17:48:33,497 INFO] Building & saving training data...  
[2020-10-02 17:54:59,490 INFO] * tgt vocab size: 50004.  
[2020-10-02 17:54:59,733 INFO] * src vocab size: 50002.  
[2020-10-02 17:55:05,870 INFO] Building & saving validation data...
```

To train the model the following command was initially used:

```
onmt_train -data data\readytotrain -save_model model\model -train_steps 50000 -gpu_ranks 0
```

```

[2020-10-02 18:02:15,788 INFO] encoder: 290090000
[2020-10-02 18:02:15,802 INFO] decoder: 55812004
[2020-10-02 18:02:15,803 INFO] * number of parameters: 84821004
[2020-10-02 18:02:15,816 INFO] Starting training on GPU: [0]
[2020-10-02 18:02:15,817 INFO] Start training loop and validate every 10000 steps...
[2020-10-02 18:02:15,818 INFO] Loading dataset from data/readyto train.train.0.pt
[2020-10-02 18:03:02,929 INFO] number of examples: 885907
Traceback (most recent call last):
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\runpy.py", line 193, in _run_module_as_main
    "main", mod_spec)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "C:\Users\Gebruiker\miniconda3\envs\nmt\Scripts\onmt_train.exe\_main_.py", line 7, in <module>
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\bin\train.py", line 197, in main
    train(opt)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\bin\train.py", line 95, in train
    single_main(opt, 0)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\train_single.py", line 150, in main
    valid_steps=opt.valid_steps)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\trainer.py", line 265, in train
    report_stats()
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\trainer.py", line 402, in _gradient_accumulation
    trunc_size=trunc_size)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\onmt\utils\loss.py", line 167, in _call_
    loss.div(float(normalization)).backward()
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\torch\tensor.py", line 185, in backward
    torch.autograd.backward(self, gradient, retain_graph, create_graph)
  File "c:\users\gebruiker\miniconda3\envs\nmt\lib\site-packages\torch\autograd\_init_.py", line 127, in backward
    allow_unreachable=True) # allow unreachable flag
RuntimeError: CUDA out of memory. Tried to allocate 392.00 MiB (GPU 0; 4.00 GiB total capacity; 2.00 GiB already allocated; 373.39 MiB free; 2.45 GiB reserved in total by PyTorch)
Exception raised from alloc at:
000007F906700000FF90670000 c10.dllic10:Error:Error [unknown file] @ [unknown line number]
000007F906700000FF90670000 c10.cuda.dllic10:CUDAOutOfMemoryError:CUDAOutOfMemoryError [unknown file] @ [unknown line number]
000007F906700000FF90670000 c10.cuda.dllic10:cuda:CUACachingAllocator::init [unknown file] @ [unknown line number]
000007F906700000FF90670000 c10.cuda.dllic10:cuda:CUACachingAllocator::init [unknown file] @ [unknown line number]
000007F906700000FF90670000 c10.cuda.dllic10:cuda:CUAStream::unpack [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cuda.dllic10:native:empty_cuda [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cuda.dllic10:native:set_storage_cuda [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cuda.dllic10:native:set_storage_cuda [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:native:mkldnn_sigmoid [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:native:mkldnn_sigmoid [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:bucketize_out [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:empty [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:native:empty_like [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:zeros_out [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:torch::functional::BatchNormFuncOptions::BatchNormFuncOptions [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:bucketize_out [unknown file] @ [unknown line number]
000007F906700000FF90670000 torch.cpu.dllic10:empty_like [unknown file] @ [unknown line number]

```

However, since there was a problem regarding the lack of video memory, an error occurred. This was fixed by modifying the command and lowering the settings:

```
onmt_train -data data\readytotrain -save_model model\model -train_steps 10000
-max_generator_batches 16 -gpu_ranks 0
```

```
(nmt) C:\Users\Gebruiker\se-csai-translation\EN-NL>nmt_train -data data\readytotrain -save_model model\model -train_steps 10000 -max_generator_batches 16 -gpu_ranks 0
[2020-10-02 18:22:33,722 INFO] * src vocab size = 50002
[2020-10-02 18:22:33,723 INFO] * tgt vocab size = 50004
[2020-10-02 18:22:33,725 INFO] Building model...
[2020-10-02 18:23:24,364 INFO] NMTModel(
  (encoder): RNNWrapper(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(50002, 500, padding_idx=1)
        )
      )
    (rnn): LSTM(500, 500, num_layers=2, dropout=0.3)
  )
  (decoder): InputFeedRNNDecoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(50004, 500, padding_idx=1)
        )
      )
    (dropout): Dropout(p=0.3, inplace=False)
    (rnn): StackedLSTM(
      (dropout): Dropout(p=0.3, inplace=False)
      (layers): ModuleList(
        (0): LSTMCell(1000, 500)
        (1): LSTMCell(500, 500)
      )
    (attn): GlobalAttention(
      (linear_in): Linear(in_features=500, out_features=500, bias=False)
      (linear_out): Linear(in_features=1000, out_features=500, bias=False)
    )
  )
  (generator): Sequential(
    (0): Linear(in_features=500, out_features=50004, bias=True)
    (1): Cast()
    (2): LogSoftmax(dim=-1)
  )
)
[2020-10-02 18:23:24,366 INFO] encoder: 29000000
[2020-10-02 18:23:24,378 INFO] decoder: 55812004
[2020-10-02 18:23:24,378 INFO] * number of parameters: 84821004
[2020-10-02 18:23:24,381 INFO] Starting training on GPU: [0]
[2020-10-02 18:23:24,381 INFO] Start training loop and validate every 10000 steps...
[2020-10-02 18:23:24,381 INFO] Loading dataset from data\readytotrain.train.0.pt
[2020-10-02 18:24:13,426 INFO] number of examples: 885907
[2020-10-02 18:24:24,289 INFO] Step 50/10000; acc: 3.99; ppl: 185048.49; xent: 12.13; lr: 1.00000; 1067/1076 tok/s; 70 sec
```

Statistics

Shown below are some of the screenshots of the results. The model took 36 minutes (2140s) to train and had a validation accuracy of 49.2999.

```
[2020-10-02 18:58:54,166 INFO] Step 9950/10000; acc: 48.28; ppl: 17.62; xent: 2.87; lr: 1.00000; 7112/7796 tok/s; 2130 sec
[2020-10-02 18:59:04,362 INFO] Step 10000/10000; acc: 47.97; ppl: 18.31; xent: 2.91; lr: 1.00000; 7596/7692 tok/s; 2140 sec
[2020-10-02 18:59:04,362 INFO] Loading dataset from data\readytotrain.valid.0.pt
[2020-10-02 18:59:04,762 INFO] number of examples: 1000
[2020-10-02 18:59:10,259 INFO] Validation perplexity: 16.2873
[2020-10-02 18:59:10,260 INFO] Validation accuracy: 49.2999
[2020-10-02 18:59:11,264 INFO] Saving checkpoint model\model_step_10000.pt
```

The model was tested using the following command:

```
nmt_translate -model model/model_step_10000.pt -src data/test.src -output
data/pred_1000_results.txt -gpu 0 -verbose
```

```
[2020-10-02 19:13:42,984 INFO]
SENT 996: ['Thank', 'you', ',', 'President-in-Office', 'of', 'the', 'Council', ',', 'for', 'attending', 'today', 'and', 'for', 'the', 'information', 'you', 'have', 'given', 'us', '.']
PRED 996: Dank u , mijnheer de fungerend voorzitter van de Raad , voor de behandeling van vandaag en voor de informatie .
PRED SCORE: -8.7845

[2020-10-02 19:13:42,984 INFO]
SENT 997: ['We', 'are', 'successful', 'internationally', 'when', 'we', 'cooperate', 'on', 'matters', 'such', 'as', 'the', 'Oil', 'Fund', ',', 'to', 'quote', 'just', 'one', 'example', '.']
PRED 997: We zijn succesvol op het gebied van internationale zaken wanneer we samenwerken op kwesties zoals de Oliefond, om te citeren, slechts één voorbeeld.
```

The model took 1.5 minute to translate and had an average score of: -0.85 as shown below.

```
[2020-10-02 19:13:42,923 INFO] PRED AVG SCORE: -0.8486, PRED PPL: 2.3364
```

Running this command outputted one file: pred_1000_results.txt. These can be compared to the test.src. The files can be found [here](#). (Angelov, D. et al., 2020).

As specified in the assignment it was recommended that BLEU was used to evaluate our model. However, when installing and running the command (as can be seen below) an error occurred. The library that was used didn't support the translation EN-NL, therefore it was decided to leave out the BLEU score until this error could be resolved.

```
(nmt) C:\Users\Gebruiker\se-csai-translation\EN-NL>pip install sacrebleu
Collecting sacrebleu
  Downloading sacrebleu-1.4.14-py3-none-any.whl (64 kB)
    |#####| 64 kB 453 kB/s
Collecting portalocker
  Downloading portalocker-2.0.0-py2.py3-none-any.whl (11 kB)
Collecting pywin32!=226; platform_system == "Windows"
  Downloading pywin32-228-cp37-cp37m-win_amd64.whl (9.1 MB)
    |#####| 9.1 MB 234 kB/s
Installing collected packages: pywin32, portalocker, sacrebleu
Successfully installed portalocker-2.0.0 pywin32-228 sacrebleu-1.4.14

(nmt) C:\Users\Gebruiker\se-csai-translation\EN-NL>sacrebleu -t wmt14 -l en-nl --echo src > wmt14-en-nl.src
Could not import signal.SIGPIPE (this is expected on Windows machines)
sacreBLEU: No such language pair "en-nl"
sacreBLEU: Available language pairs for test set "wmt14": cs-en, en-cs, de-en, en-de, en-fr, fr-en, en-hi, hi-en, en-ru, ru-en
```

References

1. Angelov, D., Cabo, Q., Franken D., Scola, S., Thissen, G. & van Wijhe, S. (2020, October) Results. Retrieved October 02, 2020, from <https://github.com/tintin10q/se-csai-translation/tree/master/EN-NL/results>
2. Tiedemann, J. (n.d.). Europarl. Retrieved October 02, 2020, from <http://opus.nlpl.eu/Europarl-v8.php>
3. Tiedemann, J. (2012, May). Parallel Data, Tools and Interfaces in OPUS. In *Lrec* (Vol. 2012, pp. 2214-2218).
4. Shterionov, D. (2020, September). Example Script. Retrieved October 02, 2020, from <https://tilburguniversity.instructure.com/courses/7304/files/629315/download>

Appendix

```
import os
import codecs
with codecs.open(os.path.join('raw_data', 'Europarl.en-nl.en'), 'r', 'utf8') as srcFile, codecs.open(os.path.join('raw_data', 'Europarl.en-nl.nl'), 'r', 'utf8') as trgFile:
    src1m = srcFile.readlines()[:1002000]
    trg1m = trgFile.readlines()[:1002000]

src = {}
trg = {}

from sklearn.model_selection import train_test_split
src['train'], src['test'], trg['train'], trg['test'] = train_test_split(src1m, trg1m, shuffle=True, test_size=2000, random_state=42)
src['dev'], src['test'], trg['dev'], trg['test'] = train_test_split(src['test'], trg['test'], shuffle=True, test_size=1000, random_state=42)

from sacremoses import MosesTokenizer
mt_src, mt_trg = MosesTokenizer(lang='en'), MosesTokenizer(lang='nl')

for split in src:
    src[split] = [mt_src.tokenize(sent.rstrip(), return_str=True, escape=False) for sent in src[split]]
    trg[split] = [mt_trg.tokenize(sent.rstrip(), return_str=True, escape=False) for sent in trg[split]]

# now let's clean the training data
src_trg_clean = [(s, t) for s, t in zip(src['train'], trg['train']) if len(s.split(' ')) + len(t.split(' ')) < 160]
src['train'] = [s for (s, t) in src_trg_clean]
trg['train'] = [t for (_, t) in src_trg_clean]

# now we are ok to save the data into a train, test and dev files
for split in src:
    with codecs.open(os.path.join('EN-NL', 'data', split + '.src'), 'w', 'utf8') as srcOutFile, codecs.open(os.path.join('EN-NL', 'data', split + '.trg'), 'w', 'utf8') as trgOutFile:
        srcOutFile.write('\n'.join(src[split]) + '\n')
        trgOutFile.write('\n'.join(trg[split]) + '\n')

# That should be it.
```

A. Code used in (train_test_dev.py) (Shterionov, D., 2020)