Under the hood of neural machine translation. Playing with pretrained models. Installation guide.

NetTT summer school
Rhodes, Greece, July 2, 2022 10:45–13:00

2022/06/07

Mikel L. Forcada
Universitat d'Alacant, Spain.
mlf@ua.es

1 Introduction

1.1 Your laptop will suffice

This hands-on section of the summer school will be devoted to playing with pretrained neural machine translation (NMT) models from the Hugging Face website. NMT needs strong computational resources, but your laptop will be enough for these experiments.

This handout shows how to install the necessary software before you come to the school and try it with English to Spanish.

If anything does not work during the installation, please make a note of (a) the step where it failed and (b) the actual error or problem found. This will improve the chances of solving the problem during the summer school.

If you would like to install other language pairs, you can download the necessary models as described in sections 5.1 and 7.

1.2 Sub-word units

The NMT systems that we will use in this school work, as most systems, work with *sub-word units*: source sentences are split into tokens which are very often smaller than words, and target sentences are also made up of the same kind of units. This gives them a chance of dealing with words not seen during training, such as inflected words, compound words, or derivatives.

We will see this with examples. The splitting and joining is performed using the SentencePiece method.

1.3 Transformers

The Hugging Face NMT systems that we will use in this school are *transformers*; this means that they produce the target sentence left to right: at each output position, the transformer computes the probability (likelihood) of each possible while *paying attention* to the tokens of the whole source sentence and to target tokens already selected; during translation, the most likely hypotheses are kept and, at the end, the one(s) with the highest total likelihood are produced. To compute these probabilities, complex multi-layer neural networks are used which contain rather complex *attention* mechanisms that select the positions of the source and target tokens which are relevant to predict the next output token.

These transformers have been trained by the Language Technology Research Group[1] at the University of Helsinki using parallel corpora available in the OPUS repository.[2] The models are available through Hugging Face.[3]

1.4 Do I need to know Python?

If you know Python, that will be helpful, but you will be able to experiment even without knowing any Python at all, as you will be able to copy and paste commands from this document.

2 Things to install before you come to the Summer School

For this session, you are expected to bring your laptop with some software installed. Some downloads are heavy; this section explains how to prepare your computer in advance. The following operating systems are covered: Ubuntu GNU/Linux (may work in similar systems), Windows 10, and MacOS.

We will assume that your laptop does not have a GPU available,[4] and we will install the CPU-only version[5] of the software.

Some downloads are quite large; this handout instructs you to install software before the school. Also, segmentation and translation models used are cached; therefore, the handout asks you to download them once.

[1] https://blogs.helsinki.fi/language-technology/

[2] https://opus.nlpl.eu/

[3] https://huggingface.co/Helsinki-NLP/

[4] GPUs or *graphics processing units* are specialized circuitry designed to efficiently perform the kinds of mathematical calculations that are necessary when training or using artificial neural networks.

[5] That is, the version that uses the regular CPU (*central processing unit*) in your laptop to execute the artificial neural network models.

2.1 Windows 10

Follow these instructions step by step:

1. Download Miniconda from https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe
2. Execute (for everyone if you are the Administrator of your laptop). This will install Miniconda, a program that allows you to install things in a particular environment without affecting the rest of things installed in your laptop.
3. Choose all the default options when installing (that is, do not change anything) and finish the installation.
4. Launch "Anaconda Prompt (Miniconda 3)". This will open a Windows command shell. Run the following commands one by one on the Windows command shell (you can copy them from this document using Ctrl-C and paste them in the shell with Ctrl-V)
5. Type
   ```
   conda deactivate
   ```
   This exits the "(base)" environment that miniconda launches by default
6. Type `cd /D %HOMEDRIVE%%HOMEPATH%`
   This takes you to your home directory.
7. Go to section 2.4.

2.2 Ubuntu GNU/Linux 20.04

1. Open a terminal (Ctrl-Alt-T works usually), download Miniconda, and launch the installer like this:
   a. `cd Downloads`[6]
   b. `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh`
   c. `chmod +x Miniconda3-latest-Linux-x86_64.sh`
   d. `./Miniconda3-latest-Linux-x86_64.sh`
   e. Scroll down and reply "yes" to the first question, confirm the location, and respond "yes" to the last question.
   f. `conda config --set auto_activate_base false`
2. Close the terminal
3. Start a new terminal
4. Type `cd`
   This takes you to your home directory
5. Go to section 2.4

[6] The name of this directory changes with the installation language: find yours.

2.3 MacOS

Installing on MacOS may be a bit trickier, as we will need two levels of virtualization, one of them (Miniconda) to have Python available, and the other one to be able to install the transformers software.

1. Open a terminal (press Command+Space Bar to launch the Spotlight bar, and type *terminal* there); then download Miniconda, and launch the installer like this:[7]
   a. `cd Downloads`[8]
   b. `curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh`[9]
   c. `chmod +x Miniconda3-latest-MacOSX-x86_64.sh`
   d. `./Miniconda3-latest-MacOSX-x86_64.sh`
   e. Scroll down and reply "yes" to the first question, confirm the location, and respond "yes" to the last question.
   f. `conda config --set auto_activate_base false`
   g. close the terminal
   h. start a new terminal
   i. type `cd`
      This takes you to your home directory
   j. proceed to the next section

2.4 Common installation steps

Perform these on your laptop before coming to the school. We will install the Transformers library. Some of the steps may not be necessary in all installations, but are included to cover as many as possible. In the terminal type

1. `mkdir summerschool`
   This creates a folder or directory called `summerschool`
2. `cd summerschool`
   This takes us to that directory; files created will be placed here
3. `conda create -n testing`
   This creates a new environment called testing where our software will be installed. Reply "yes" when prompted.
4. `conda activate testing`
   This activates the environment so that all we install from now on will be installed only in that environment; answer "y" to the question.

[7] Newer ARM-based Macs may need to download a different version of Miniconda, namely https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-arm64.sh ; this implements an older version of Conda and Python. I have not had time to test this course there.

[8] The name of this directory changes with the installation language: find yours.

[9] On newer Macs with ARM processors, try https://repo.anaconda.com/miniconda/Miniconda3-py39_4.11.0-MacOSX-arm64.sh

5. `conda install python`
   This will install a recent version of the Python language and a number of related tools. Answer “y” to the last question. This may take a while.
6. **[Needed for MacOS only]** `python -m venv .env`
   This creates another virtual environment (`transformers` does not install on conda alone on MacOS)
7. **[Needed for MacOS only]** `source .env/bin/activate`
   This activates the virtual environment `.env`.
8. `pip install transformers[torch]`
   This installs a version of the `transformers` neural network software package needed to run the models we will download from Hugging Face. We install a version that works without a GPU and uses the PyTorch package. This is another large download (around 1 GB); be patient.
9. `pip install sacremoses`
10. `pip install sentencepiece`
    These last two steps install the SentencePiece software that deals with the segmentation (tokenization) of text into tokens (words or sub-word units).
11. `pip install jupyter`
    This installs the *Jupyter* software which will allow running Python commands from a web-based editable document where inputs and outputs are shown together, and where any operation can be repeated. All Python commands in the Jupyter notebook that we will use in the course can also be run on the interactive Python shell.

3 A translation model

To start, we will install an English→Spanish transformer model[10] from the HelsinkiNLP set in Hugging Face.[11]

This model has been trained using a parallel corpus assembled from various parallel corpora[12] from OPUS[13] and tuned to the Tatoeba corpus (a conversational corpus) there. The training corpus has about 150.000.000 sentence pairs, and the development (tuning) corpus has about 200.000 sentence pairs.

The model is a BART-style transformer model with a joint vocabulary of 65000 sub-word units, a 5-layer encoder and a 5-layer decoder with 512 units in each layer.

The tokenizer segmenting the sentence in sub-word units is based on SentencePiece.

[10] https://huggingface.co/Helsinki-NLP/opus-mt-en-es

[11] https://huggingface.co/Helsinki-NLP

[12] Books, DGT, ECB, ELRA-W0147, EMEA, EUbookshop, EUconst, Europarl, GlobalVoices, GNOME, JRC-Acquis, JW300, KDE4, MultiUN, News-Commentary, OpenSubtitles, ParaCrawl, PHP, QED, SciELO, Tanzil, TED2013, TildeMODEL, UN, UNPC.

[13] http://opus.nlpl.eu

3.1 Preinstalling the model

In the command shell (terminal) of your operating system, type `python` (if you started a new command shell, be sure to activate the virtual environment by typing `cd summerschool`, `conda activate testing`, and, if in MacOS, `source .env/bin/activate ).` This launches the python command shell, which will greet us with a few messages about its version, the local operating system, etc, and finally, “`>>>`”. Now, we type python commands, and they will be executed. The examples here are for English-to-Spanish. Here’s what we will type:

1. `from transformers import AutoTokenizer, AutoModelForSeq2SeqLM`
   This downloads specific parts of the transformer model, in particular, the Autotokenizer (which will take care of turning sentences into sequences of sub-words) and AutoModelForSeq2SeqLM (which will install the transformer neural network model that transforms sequences of source-side sub-words into target-side sub-words).
2. `tokenizer = AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-en-es")`
   This downloads from Hugging Face the English-to-Spanish tokenizer packages pretrained on OPUS parallel corpora. It may take a while. Tokenizers are specific for each language pair.
3. `model = AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-en-es")`
   This downloads from Hugging Face the English-to-Spanish model, that is, the neural machine translation system itself. This is a large download, around 300 MB.
4. Exit the Python shell now with `exit()`, and the operating system shell (or, in Windows, the Anaconda prompt shell). Everything we need for the first round of tests is installed. Or you may continue in the next section.

If you want to prepare your laptop to run other machine translation models instead (there are many available in the Helsinki-NLP set of Hugging Face, all trained on OPUS corpora. The models we are interested in are in URL https://huggingface.co/Helsinki-NLP and have names with the form `opus-mt-`*X*`-`*Y* where *X* and *Y* are usually language codes.[14]

You may, for instance, be interested in Greek to English. It’s quite straightforward. If you have exited it, fire up a terminal, launch your virtual environment(s) as explained above, launch the Python shell and then paste the following commands:

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
tokenizer = AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-grk-en")
model = AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-grk-en")
```

If you want to experiment with other language pairs, it might be a good idea to *preload* them in the same way. One way to get very detailed information about the neural architecture of each model is to execute `model.cpu()`.

[14] Some models group languages, such as `Helsinki-NLP/opus-mt-ROMANCE-en.`

4 Testing that it works before coming to the summer school

Once you have performed the steps before, we can test if everything has been correctly installed. If you have closed the python command shell and the command shell of your operating system, please be sure to take the appropriate steps for your operating system below.

4.1 Windows

1. In Windows Launch "Anaconda Prompt (Miniconda 3)"; in the shell, type:
   - `conda deactivate`
2. Then type
   - `cd /D %HOMEDRIVE%%HOMEPATH%`
   - `cd summerschool`
   - `conda activate testing`
   - `python`

4.2 Ubuntu GNU/Linux 20.04

1. Start a new terminal (for instance, pressing Ctrl-Alt-T)
2. Type
   a. `cd`
   b. `cd summerschool`
   c. `conda activate testing`
   d. `python`
3. go to Section 4.4.

4.3 MacOS

1. Open a terminal (press Command+Space Bar to launch the Spotlight bar, and type *terminal* there).
2. Then type
   a. `cd`
   b. `cd summerschool`
   c. `conda activate testing`
   d. `source .env/bin/activate`
   e. `python`
3. proceed to the next section

4.4 Common to all three operating systems

Execute the Python commands in section 3.1 (corresponding to English to Spanish) again. You will see that the models do not take as long as before to download, as they are “cached” (stored) from the last download:

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
```

```
tokenizer = AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-en-es")
```

```
model = AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-en-es")
```

Now, we will issue a simple command to test that everything works fin. Paste it on the interactive Python shell:

```
print(tokenizer.decode(model.generate(tokenizer.encode("This is a test",
return_tensors="pt", max_length=512,
truncation=True))[0],skip_special_tokens=True))
```

If you get “*Esto es una prueba.*” everything works fine (we will study this in detail during the summer course). Exit the Python shell with

```
exit()
```

If you are on a Mac, exit the second virtual shell

```
source deactivate
```

And, on all computers,

```
conda deactivate
```

You’re good to go! Safe travels! See you in Rhodes!