

Practical work on Statistical MT with Moses

EPFL HLT Course, November 2016 – Andrei Popescu-Belis

Version 2 – with Virtual Box and pre-installed Moses

0. Overview

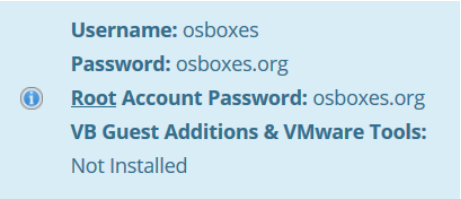
The goal is to experiment with the Moses phrase-based SMT system, then to build a translation model in a language pair of your choice, a language model in the target language, and finally run Moses in order to translate new sentences. Firstly this should be done with the simplest data and options, then gradually you can try to change things in order to improve the output.

For **documentation**: see <http://www.statmt.org/moses/>, but it is also possible to get an offline PDF version at <http://www.statmt.org/moses/manual/manual.pdf>. Overview of what to do:

1. **Install VirtualBox** (from Oracle), get a copy of Debian 8.5 (64bit).vdi image from APB, create virtual machine on your laptop and launch it.
2. For **simple testing**, follow the instructions for the decoding experiments in “*Phrase-based Tutorial*” at <http://www.statmt.org/moses/?n=Moses.Tutorial> (up to “Verbose”).
3. For **training new translation and language models** and running Moses, follow the instructions on “*Baseline System*” in the PDF manual or at www.statmt.org/moses/?n=Moses.Baseline.

1. Install Virtual Box, obtain image with Moses and create your VM

- Download and install Oracle VirtualBox from <https://www.virtualbox.org/wiki/Downloads>.
- Obtain a copy of the *Debian 8.5 (64bit).vdi* file from APB¹. This will allow you to execute this virtual machine on your system (64 bit) which comes with Debian OS and Moses files.
- To create the virtual machine with the VDI file, follow the instructions at <http://www.osboxes.org/guide/> (or the virtualbox.org manual).
- Launch the virtual machine and log in.
- There are possible keyboard issues (QWERTZ vs. QWERTY): select the language when you log in at the upper right corner, then open the preferences (also from upper right) and select input language and source as “English” if you have a US keyboard (the VBI image has a Swiss keyboard).



Username: osboxes
Password: osboxes.org
Root Account Password: osboxes.org
VB Guest Additions & VMware Tools:
Not Installed

The Moses system is installed under `/opt/moses/`. The home directory is `/home/osboxes/`. The data and system configuration file are under `/home/osboxes/MT/`. This contains a file called **apb-commands.txt** which has a copy of the commands to be executed, with the correct paths if you work under MT.

¹ This is based on the Debian image from <http://www.osboxes.org/debian/>, to which I added Moses provided as a Debian package (at <http://www.statmt.org/moses/?n=Moses.Packages>, specifically “RELEASE-2.1 Debian”), and then I prepared data and trained models based on Moses tutorials.

2. Test Moses with the provided TM and LM

Following the instructions at <http://www.statmt.org/moses/?n=Moses.Tutorial> from the start.

Get (small) models from <http://www.statmt.org/moses/download/sample-models.tgz>, unpack them.
Or use directly the files in `/home/osboxes/MT/tutorial/sample-models/`

Run Moses as indicated in the tutorial (either command line or with in/out files).

Try some sentences in German and press Enter: how well does it translate? Note that for this test TM and LM, the German and English vocabulary is extremely small. Take a look at `phrase-model/phrase-table` (it's a readable text file) and count how many different words and phrases are stored. Try to vary your input sentences and see what happens.

The following steps can also be tried: the "trace" and the "verbose" options

3. Learn a new TM and LM, use them to translate new sentences

Create a folder for your new experiments, follow <http://www.statmt.org/moses/?n=Moses.Baseline>, starting from "Corpus preparation" (GIZA++ is installed).

Get parallel data, either as suggested in the tutorial (News Commentary v8 from WMT13), or from: <http://opus.lingfil.uu.se/>. Start with the smallest possible corpus (for instance the 5000 first lines of NCv8 or a small corpus such as <http://opus.lingfil.uu.se/RF.php> which has only 151 sentences) in order to check the entire processing chain without waiting too much. If you use Opus, make sure you download the right format ("*download plain text files (MOSES/GIZA++)*") for the language pair of your choice. For instance, try EN/FR or EN/DE.

- Follow the instructions at <http://www.statmt.org/moses/?n=Moses.Baseline> for **corpus preparation**. Use the tools to perform *tokenization*, *true casing*, and *pruning of long sentences*.
 - For instance, use the commands in `apb-commands.txt` to perform the above operations, then use "`head -5000`" to extract the first 5000 lines of the source and target files of the NCv8 corpus..
 - Note that in the `/MT/working.5000` folder provided with the image, there is already a trained model based on these files.
- Learn a **language model** as instructed in the tutorial (with KenLM). Binarize it. Test the LM with the "`query`" command as explained in the tutorial.
- Learn a **translation model** with "`/opt/moses/scripts/training/train-model.perl`" as explained in the tutorial. This is very fast for 150 sentences, but gives a very poor model. The larger the corpus, the longer the training time, but the better the model. (Note that in this TP we will not be able to binarize the models, which saves space, because the function to do it is not available in the Debian packages).
- **Translate some new sentences**. Look at the training corpus so that you only use words that are known to the model (pay attention to their capitals too). How good are the translations?

Optional: tune the weights of the factors of the Moses MT decoder. Get a new small parallel corpus in the same language to **perform tuning of the weights** in `moses.ini`, as explained at

<http://www.statmt.org/moses/?n=Moses.Baseline>, section on *Tuning*. Tuning is typically the longest training stage because it involves translating the tuning set multiple times. Hence a tuning set much smaller than the training set is typically used.

4. Evaluating the results

Provide a test corpus (one sentence per line) to Moses and **evaluate the quality of the automatic translations** by measuring the BLEU score of the translated corpus. Use the implementation of the BLEU metric provided as `/opt/moses/scripts/generic/multi-bleu.perl` – its arguments are one or more reference translations, one sentence per line, and the candidate translation (to be evaluated) is provided as an incoming stream (with '< filename'). The closer the domains of the training and test corpora are, the higher the BLEU score should be (typically in the 10-30% range). Testing on the training corpus should provide an unusually high BLEU score (e.g. 80-90%).

5. Extensions

5.1. New language pair with more data. You can use a language pair of your choice for which you find parallel data at <http://opus.lingfil.uu.se/>. Try now to learn a better translation model (using more data and tuning) and check your BLEU score. Don't start too large: try a factor 10 or 100 every time you increase the data size to find out how long the entire process takes. Repeat the process with improved parameters (TM, LM, Moses options) and try to improve your BLEU score on your test data.

5.2 Streamlining the training process. Use the *Experiment Management System* (described in Sections [2.6.8](#) and [3.4](#) of the Moses manual) to simplify pre-processing and training (not tested on Cygwin). You will need to train the LM separately.

5.3 One possible idea for a course project. Build an MT system and test it on WMT 2011 data. Use training and tuning data as provided, but try to test only once (do not optimize on test data). Try to modify some advanced decoding parameters as described in [Section 3.3](#) of the manual or online at [Optimizing Moses](#). Training data (amount and domain) is still essential, as is computing power to speed up tuning and then decoding (consider the multi-threaded version). How does your BLEU score compare to the published results?

5.4 Another possible idea for a course project. Combine the SMT decoder with Lucene to perform cross-lingual just-in-time information retrieval in another language. For instance, while you write in your native language, get results from English Wikipedia.