Lab on PB-SMT

Machine Translation – Summer 2018

May 29, 2018

1 Getting Started

The purpose of this lab is to train and evaluate a phrase-based statistical machine translation system (PB-SMT) based on Moses[1,2].

First of all you need to install the main software. The process is well documented and there is a very active mailing list where most of the possible installation issues have already been solved by the Moses' developers:
`http://www.statmt.org/moses/?n=Development.GetStarted`
`http://www.statmt.org/moses/?n=Moses.MailingLists`

1.1 Modules

You do not need to install all the components for this exercise, but be sure you have a tool for each module:

Language Model We will use `KENLM` or `SRILM`. `KENLM` is installed with Moses by default but you have to fill a form to download `SRILM`:
`http://www.speech.sri.com/projects/srilm/download.html`

Word Alignment We will use either `GIZA++` or `mgiza`
`https://github.com/moses-smt/giza-pp`
`https://github.com/moses-smt/mgiza`

Translation Model, Decoder and Processing Scripts All within `Moses`:
`https://github.com/moses-smt/mosesdecoder`

1.2 Tips and Advises

- Do not use Windows
- Read the full installation instructions before starting

2 (Statistical) Translation Engine for Literature

2.1 Getting the data

Parallel corpora can be found in evaluation campaigns' web sites (WMT, NIST, IWSLT, etc.) as we have seen in the lectures or the open parallel corpus OPUS site. Explore the latter:

`http://opus.lingfil.uu.se/`

For this session, we will use the English–French edition of the *Books collection*, a set of copyright-free books aligned by Andras Farkas. Download them (download plain text files, MOSES/GIZA++), extract the files and explore the content:

[1] `http://www.statmt.org/moses/`
[2] `https://github.com/moses-smt/mosesdecoder`

```
machine:~/pln/teaching/literature$ wc -lw Books.en-fr.*
  127085  2715071 Books.en-fr.en
  127085  2642300 Books.en-fr.fr
  127085   531211 Books.en-fr.ids

machine:~/pln/teaching/literature$ tail -n 2 Books.en-fr.*
==> Books.en-fr.en <==
The corpses lay all night, spread out contorted, on the dining-room floor, ...
And for nearly twelve hours, in fact until the following day at about noon, Madame Raquin, ...

==> Books.en-fr.fr <==
Les cadavres restèrent toute la nuit sur le carreau de la salle et manger, ...
Et, pendant près de douze heures, jusqu'au lendemain vers midi, Mme Raquin, ...

==> Books.en-fr.ids <==
en/Zola_Emile-Therese_Raquin.xml.gz fr/Zola_Emile-Therese_Raquin.xml.gz s1246.0 s1246.0
en/Zola_Emile-Therese_Raquin.xml.gz fr/Zola_Emile-Therese_Raquin.xml.gz s1246.1 s1246.1
```

The ids file tells us to which novel corresponds each parallel fragment. Use this information to extract all the fragments corresponding to *The Great Shadow* by Arthur Conan Doyle. We will use this novel for in-domain tuning and testing purposes (top 1000 lines for testing, the remaining bottom ones for tunning). All the other novels (remove this one!) are used for training.

```
machine:~/pln/teaching/SMT/corpus$ grep Shadow Books.en-fr.ids | wc -lw
   1857    7921

machine:~/pln/teaching/SMT/corpus$ wc -lw Books.*.??
  125226  2674214 Books.train.en
  125226  2600120 Books.train.fr
    1000    22930 Books.test.en
    1000    23551 Books.test.fr
     859    17927 Books.dev.en
     859    18629 Books.dev.fr
```

Let's go back to OPUS and download also *WMT-News*, a parallel corpus of News Test Sets provided by WMT for testing purposes. The last 3002 lines correspond to the test set of 2014 (the last year that included the English–French pair). We will use this data to test our system in an out-of-domain framework.

```
machine:~/pln/teaching/SMT/corpus$ wc -lw news2014.test.*
  3002  62333 news2014.test.en
  3002  68148 news2014.test.fr
```

2.2 Pre-processing the Data

This is probably the most important part of the process that fully depends on you. Look carefully at the data. Which is the encoding of the file? Text is homogeneous? Same punctuation marks? Strange characters? In our case, we are using data that has already been prepared for MT, we are in a controlled experiment, but this is not the usual case. Get used to doing the whole process!

We are using Moses, which has its own scripts for **cleaning** the texts. Look at the folder `mosesdecoder/scripts/tokenizer`. Using the next scripts never harms:

```
 perl replace-unicode-punctuation.perl < set.en > set.norm1.en
 perl normalize-punctuation.perl -l en < set.norm1.en > set.norm2.en
 perl remove-non-printing-char.perl < set.norm3.en > set.norm3.en
```

where *set* applies to the training, development and the two test sets.

The final file is ready to be **tokenised**:

```
perl tokenizer.perl -l en -no-escape -threads 4 < set.norm3.en > set.tok.en
```

Optionally, we can **truecase** the text or, at least, lowercase it. Moses also has tools for training a truecaser and a recaser. Look at the folder `mosesdecoder/scripts/recaser`.

```
perl train-truecaser.perl --model tcModel.Books.en --corpus Books.train.tok.en
```

Unless there is a reason, you should train the truecaser with large monolingual corpora. Here, we will only use the two sides of our parallel corpus. Look at the model: it is a frequentist list of the tokens with their capitalisation in our corpus:

```
machine:~/pln/teaching/SMT/corpus/truecaser$ more tcModel.Books.en
  patronne (1/1)
  Parvis (24/28) parvis (4)
  quarreling (6/6)
  ...
```

After training one model for each language, we want to apply it to all the tokenised sets:

```
perl truecase.perl --model ./truecaser/tcModel.Books.en < set.tok.en > set.tc.en
```

And we are done with the pre-processing. This is the corpus we are going to use for training, tuning and testing our translation engine:

```
machine:~/pln/teaching/SMT/corpus/truecaser$  ls *tc*
  Books.dev.tc.en  Books.test.tc.en  Books.train.tc.en  news2014.test.tc.en
  Books.dev.tc.fr  Books.test.tc.fr  Books.train.tc.fr  news2014.test.tc.fr
```

2.3 Language Modeling

Multiple language modeling toolkits are available for language model (LM) estimation, statistical, neural, hadoop-based... `KENLM` is installed by default by Moses and it is good for most of the standard applications. Use one the others (`SRILM`, `IRSTLM`, `RandLM`, `NPLM`, `RDLM`...) if you have any special need though (huge corpora, interpolations, etc.).

At the end, most of the language models can be used within Moses because they share a standard format, the ARPA format:

```
 \data\
  ngram 1=n1
  ngram 2=n2
  ...
  ngram N=nN

  \1-grams:
  p w [bow]
  ...

  \2-grams:
  p w1 w2 [bow]
  ...

  \N-grams:
  p w1 ... wN
  ...

\end\
```

Let's estimate a LM with `KENLM`. It applies a modified Kneser-Ney smoothing and no pruning. Pass the order of the LM via the -o argument, an amount of memory to use for building (-S), and a location to place temporary files (-T).

```
lmplz -o 5 -S 80% -T ./tmp <../corpus/Books.train.tc.en > corpusBooks.train.tc.5.en.lm
```

```
\data\
ngram 1=69286
ngram 2=700890
ngram 3=1807372
ngram 4=2550220
ngram 5=2779881

\1-grams:
-5.875923 <unk>0
0 <s>-1.4962252
-2.9368296 </s>0
-2.1009169 the -0.66151506
-5.737095 Wanderer -0.12696436
...
-0.6858582 Therese explained the bruises disfiguring
-1.1650921 , annoyed at being forestalled
-1.3933828 forestalled , began to declaim

\end\
```

2.4 Training

Now we have all the tools for training a PB-SMT system. Moses has a script to run (almost) all the training steps:

```
 0. Prepare corpus
 1. Prepare data for GIZA                (45 minutes) -> (3 minutes)
 2. Run GIZA++                             (16 hours) -> (30 minutes)
 3. Align words                          (2:30 hours) -> (15 seconds)
 4. Get lexical translation table        (30 minutes) -> (15 seconds)
 5. Extract phrases                      (10 minutes) -> (1 minute)
 6. Score phrases                        (1:15 hours) -> (3 minutes)
 7. Build lexicalized reordering model       (1 hour) -> (45 seconds)
 8. Build generation models                     -         -
 9. Create configuration file              (1 second) -> (1 second)
10. Log-linear model tuning parameters,               -> (90 minutes)
    (MERT)
```

Visit `http://www.statmt.org/moses/?n=FactoredTraining.HomePage` for a summary. The run times within the first parentheses refer to a training run on a 751,000 sentence, 16 million word German–English Europarl corpus, on a 3GHz Linux machine. A state-of-the-art German–English corpus has more than 5 million parallel fragments, for French–English one can gather more than 20 million parallel fragments. For this exercise, we are using 125,000 fragments so that the training time (and translation quality) is considerably reduced. Notice that using `mgiza` which allows multi-threading and/or the `--parallel` option further reduces the training time. The run times within the second parentheses refer to our system using the `--parallel` option on a 2.6GHz Linux machine.

Explore the folder `mosesdecoder/scripts/training`. Here you have all the scripts needed for training your system. We start with `clean-corpus-n.perl`. `GIZA` does not properly deal with long sentences, so the first thing we do is further cleaning our Books.train.tc corpus by removing sentences longer than 100 tokens, and sentence pairs with a large length difference (the default length ratio is 9).

```
perl clean-corpus-n.perl
     ../corpus/Books.train.tc en fr ../corpus/Books.train.tc.clean 1 100
```

This version of the corpus can be used within the training pipeline. The `train-model.perl` script runs steps 1 to 9, although a subset of them can be run with the `--first-step --last-step` arguments. From here on, execute everything twice to obtain an en2fr and a fr2en translation engine.

```
 perl train-model.perl   --parallel   -root-dir ./   -f fr   -e en
      -corpus /home/cristinae/pln/teaching/SMT/corpus/Books.train.tc.clean
      -alignment grow-diag-final-and    -reordering msd-bidirectional-fe
      -lm 0:5:/home/cristinae/pln/teaching/SMT/lm/corpusBooks.train.tc.5.en.lm:8
      --external-bin-dir /home/cristinae/soft/mosesdecoder/bin
```

Add `-mgiza -mgiza-cpus 4` or whatever number of CPUs you have if you use `mgiza`. The option `--score-options '--GoodTuring'` is also recommended.

At this point, the training is complete but we still need to tune the parameters of the log-linear model. This is clearly shown in the configuration file (fr2en):

```
#########################
### MOSES CONFIG FILE ###
#########################

# input factors
[input-factors]
0

# mapping steps
[mapping]
0 T 0

[distortion-limit]
6

# feature functions
[feature]
UnknownWordPenalty
WordPenalty
PhrasePenalty
PhraseDictionaryMemory name=TranslationModel0 num-features=4
       path=/..../phrase-table.gz input-factor=0 output-factor=0
LexicalReordering name=LexicalReordering0 num-features=6
       type=wbe-msd-bidirectional-fe-allff input-factor=0 output-factor=0
       path=/..../reordering-table.wbe-msd-bidirectional-fe.gz
Distortion
KENLM name=LM0 factor=0 path=/..../corpusBooks.train.tc.5.en.lm order=5

# dense weights for feature functions
[weight]
# The default weights are NOT optimized for translation quality. You MUST tune the weights.
# Documentation for tuning is here: http://www.statmt.org/moses/?n=FactoredTraining.Tuning
UnknownWordPenalty0= 1
WordPenalty0= -1
PhrasePenalty0= 0.2
TranslationModel0= 0.2 0.2 0.2 0.2
LexicalReordering0= 0.3 0.3 0.3 0.3 0.3 0.3
Distortion0= 0.3
LM0= 0.5
```

Before tuning the weights, let's stop and look at all the files that have been generated up to now. Four folders have been created: `corpus`, `giza.en-fr`, `giza.fr-en` and `model`. The first three are inputs/outputs for the word alignments. Let's comment file per file. The input for `GIZA` is in `corpus`:

```
machine:~/pln/teaching/SMT/moses/corpus$ head -5 *.vcb *snt
```

```
==> en.vcb <==
1 UNK 0
2 ,203899
3 the 137305
4 . 112825
5 " 73243

==> fr.vcb <==
1 UNK 0
2 ,212046
3 . 109515
4 de 92960
5 la 55390

==> en-fr-int-train.snt <==
1
7 164 682
3 59761
1
74689

==> fr-en-int-train.snt <==
1
3 59761
7 164 682
1
65229
```

And the aligments in giza.en-fr (and giza.fr-en):

```
machine:~/pln/teaching/SMT/moses/giza.en-fr$ zmore en-fr.A3.final.gz

# Sentence pair (1) source length 3 target length 2 alignment score : 0.000188732
the Wanderer
NULL ({ }) le ({ 1 }) grand ({ }) Meaulnes ({ 2 })

# Sentence pair (2) source length 1 target length 1 alignment score : 0.525664
Alain-Fournier
NULL ({ }) Alain-Fournier ({ 1 })

# Sentence pair (3) source length 2 target length 2 alignment score : 0.0198902
first Part
NULL ({ }) première ({ 1 }) PARTIE ({ 2 })
```

Notice that up to this point everything is symmetric, so if you want to train a system on the other direction you can skip steps 1 to 3 and use the same three folders. Finally, the output of GIZA is post-processed to extract phrase alignments, reorderings and probabilities, and the corresponding files are stored in model:

```
machine:~/pln/teaching/SMT/moses/model$ head -3 *

==> aligned.grow-diag-final-and <==
0-0 1-1 2-1
0-0
0-0 1-1

==> extract.inv.sorted <==
! ! ! ||| ! ||| 0-0 1-0 2-0
! " " ’ - ||| ! ... " A opéré ||| 0-0 1-1 1-2
! " " ’ - ||| ! ... " A ||| 0-0 1-1 1-2
```

```
==> extract.o.sorted <==
! ! ! ! " hurla le ||| ! ’ yelled the ||| other mono
! ! ! ! " hurla ||| ! ’ yelled ||| other mono
! ! ! ! " ||| ! ’ ||| other mono

==> extract.sorted <==
! ! ! ! " hurla le ||| ! ’ yelled the ||| 0-0 1-0 2-0 3-0 4-1 5-2 6-3
! ! ! ! " hurla ||| ! ’ yelled ||| 0-0 1-0 2-0 3-0 4-1 5-2
! ! ! ! " ||| ! ’ ||| 0-0 1-0 2-0 3-0 4-1

==> lex.e2f <==
indépendantes independent 0.0606061
indépendantes disconnected 0.0833333
indépendantes unscrupulous 0.1250000

==> lex.f2e <==
independent indépendantes 0.5000000
disconnected indépendantes 0.2500000
unscrupulous indépendantes 0.2500000

==> phrase-table <==
! ! ! ! " hurla le ||| ! ’ yelled the ||| 1 0.0021 1 0.0039 ||| 0-0 1-0 2-0 3-0 4-1 5-2 6-3 ||| 1 1 1
! ! ! ! " hurla ||| ! ’ yelled ||| 1 0.0105226 1 0.00682231 ||| 0-0 1-0 2-0 3-0 4-1 5-2 ||| 1 1 1 |||
! ! ! ! " ||| ! ’ ||| 0.00154799 0.0435936 1 0.0438577 ||| 0-0 1-0 2-0 3-0 4-1 ||| 646 1 1 ||| |||

==> reordering-table.wbe-msd-bidirectional-fe <==
! ! ! ! " hurla le ||| ! ’ yelled the ||| 0.2 0.2 0.6 0.6 0.2 0.2
! ! ! ! " hurla ||| ! ’ yelled ||| 0.2 0.2 0.6 0.6 0.2 0.2
! ! ! ! " ||| ! ’ ||| 0.2 0.2 0.6 0.6 0.2 0.2
```

For translating new sentences with `Moses`, you need the translation model (`phrase-table`), the reordering model (`reordering-table.wbe-msd-bidirectional-fe`), the language model you have estimated before (`corpusBooks.train.tc.5.en.lm`) and the configuration file (`moses.ini`) with an appropriate weight for each feature. To estimate the weigths, `Moses` implements several algorithms (`MERT`, `MIRA`, `PRO`...). Use MERT for the optimisation:

```
 mert-moses.pl /home/cristinae/pln/teaching/SMT/corpus/Books.dev.tc.fr   \
               /home/cristinae/pln/teaching/SMT/corpus/Books.dev.tc.en   \
               /home/cristinae/soft/mosesdecoder/bin/moses    ./model/moses.ini  \
               --mertdir /home/cristinae/soft/mosesdecoder/bin/    \
               --working-dir=/home/cristinae/pln/teaching/SMT/moses/tuning
```

We have named the output directory `tuning`. Once the optimisation has finished, explore the folder, see how BLEU on the development set evolves from iteration to iteration (`runX.moses.ini` contains this information), and look at the final `moses.ini`. This file is what you need for translating.

2.5 Decoding

If you look at the size of the models you need for decoding, you can see that they are moderately big:

```
-rw-r--r-- 1  208M Jun 19 10:22 phrase-table.gz
-rw-r--r-- 1   80M Jun 19 10:23 reordering-table.wbe-msd-bidirectional-fe.gz
-rw-r--r-- 1  298M Jun 18 13:18 corpusBooks.train.tc.5.en.lm
```

But they can be much bigger! Observe the size of the models obtained from an English–French parallel corpus with ~ 20 M parallel sentences:

```
-rw-r--r-- 1   20G Jun 13 13:23 phrase-table.gz
-rw-r--r-- 1  7,2G Jun 13  2016 reordering-table.wbe-msd-bidirectional-fe.gz
-rw-r--r-- 1   20G Jun 18  2016 genBioWP.tc.en.5.lm
```

Since these models are loaded into memory, it is common practice to filter and binarise them. For instance, before translating into English the French test set with the Conan Doyle novel, run:

```
perl filter-model-given-input.pl    ./filteredBooks   \
       /home/cristinae/pln/teaching/SMT/moses/tuning/moses.ini   \
       /home/cristinae/pln/teaching/SMT/corpus/Books.test.tc.fr
```

And then, use the new configuration file with the filtered models to run the decoder:

```
moses -f filteredBooks/moses.ini  < ../corpus/Books.test.tc.fr
         > trads/Books.testTrad.tc.fr2en.en
```

Do the same for the news2014 test set. At this point you should have the translations ready:

```
machine:~/pln/teaching/SMT/moses/trads$ wc -l *
   1000 Books.testTrad.tc.fr2en.en
      0 filteredBooks
      0 filteredNews
   3002 news2014.testTrad.tc.fr2en.en
```

2.6 Evaluation

The last point of the process is the evaluation of your system. This is an important aspect very difficult in MT. Nothing can really substitute a manual evaluation of your translations, but this is a very expensive evaluation (both in time and money) that cannot be used during the development of a system. Instead, automatic evaluation is used. In order to have a first insight on the quality of your engine, let's estimate BLEU with another `Moses` script:

```
perl /home/cristinae/soft/mosesdecoder/scripts/generic/multi-bleu.perl
usage: multi-bleu.pl [-lc] reference < hypothesis
Reads the references from reference or reference0, reference1, ...

perl /home/cristinae/soft/mosesdecoder/scripts/generic/multi-bleu.perl  \
     ../corpus/Books.test.tc.fr < trads/Books.testTrad.tc.fr2en.en

BLEU = 19.51, 55.8/26.0/13.5/7.4 (BP=1.000, ratio=1.000, hyp_len=26610, ref_len=26601)
```

Your engine has been optimised for translating novels. Since the corpus is small and only this kind of text has been used for training, other genres will have really low quality. Using a standard test set such as the news2014 one allows you to compare with the best system on that data. Look at the best system here:

`http://matrix.statmt.org/matrix`

If you can beat that score, your system is really good!

However, keep in mind that BLEU is only a lexical metric that counts the number of n-gram matches. Two translations can be perfect, or at least convey the same meaning, without having any n-gram in common. As advice, if you can only afford automatic evaluations, do not use only BLEU: try to combine lexical, syntactic and semantinc metrics whenever is possible. This will make your conclusions stronger. We will devote one lecture to further explore this.

3 After the Session

Toy vs. Real Engines During this lab you have built a toy SMT system. The only reason it is a *toy* system is the size of the corpus. You can build a real engine by using larger parallel corpora as that given for the MT evaluation campaigns:

`http://www.statmt.org/wmt17/translation-task.html`

Pivot Translation A good way to check that you have understood what is under the hood is to do translation via a *pivot* language. Let's assume we have data for training an English-to-Esperanto system and a German-to-English system, but we have no parallel corpora German–Esperanto. You can always do a translation pipeline German-to-English → English-to-Esperanto, but a more clever (and effective) way to go is through a phrase table combination.

If you have understood the content of the Moses' configuration file and how translation tables work, you should be able to train a translation system via pivoting, that is, via a phrase table combination. The following article explains the basic ideas which are implemented in the `TmTriangulate` tool:

- Zhu, Xiaoning, Zhongjun He, Hua Wu, Conghui Zhu, Haifeng Wang and Tiejun Zhao. *Improving Pivot-Based Statistical Machine Translation by Pivoting the Co-occurrence Count of Phrase Pairs.* EMNLP (2014).
- `https://github.com/tamhd/MultiMT`

Give it a try if you want to consolidate your learning.

4 Questionnaire

1. Look for the following sentence in your training corpus and write its output after each one of your pre-processing steps:
`When he was gone, I turned to the boy, whom they called Xury, and said to him, "Xury, if you will be faithful to me, I'll make you a great man; but if you will not stroke your face to be true to me"-that is, swear by Mahomet and his father's beard-"I must throw you into the sea too."`

2. What is the modified Kneser-Ney smoothing you have used for estimating the language model?

3. Look for the word `beard` in your language model. Write three lines of your language model that include n-grams containing this word (three examples with 1-grams, three with 2-grams, ..., three with 5-grams). What represent the scores before and after the n-grams? Which of the examples you show is more probable in English?

   Example 4-gram: `-0.33215663 beard flying in the -0.017270327`

4. What is the Good-Turing method you may have used for the training?

5. Look for the sentence of Question 1 in GIZA's output. To which word is aligned `beard`? Find an example of a bad word aligment in that sentence (it's a long sentence, there are lots of them!). Go to the equivalent sentence in French (the other A3 file) and look for the alignment of `barbe`. To what token is aligned now the mistake you have found in the English alignment? Find the same sentence in the symmetric alignment file and comment the differences.

6. What does the heuristic grow-diag-final-and? Which is its purpose?

7. Let's look now into the translation table. Write down the entries for `beard` and for `black beard`. What's the meaning of all the scores available for each translation option? Look for the same entries in the reordering model and write them down.

8. We have used MERT for tuning the model parameters, but Moses implements more algorithms. Explain briefly the main idea behind MIRA and compare it to MERT.

9. Let's look into MERT training. How many iterations did it run? Write the weights obtained after convergence and compare them to those in run 3. Plot the evolution of BLEU through iterations. Does the convergence point correspond to the best BLEU point?

10. Translate *Books.test.en* and *news2014.test.en* into French and *Books.test.fr* and *news2014.test.fr* into English (you need to go through the truecased version because it is how the systems have been trained). Write the BLEU scores for these translations. Compare the BLEU scores for news2014 with the best system at `http://matrix.statmt.org/matrix`. Compare also some translated sentences from your *news2014.test.en2fr* to those in the best phrase-based SMT system (uedin-wmt14-en-fr) and to those in a rule-based system (PROMT Rule-based). You can find their outputs in `http://matrix.statmt.org/matrix/systems_list/1747`. Try to reason the comparison according to the size of the corpus used for training and the differences between rule-based and statistical systems.