

EXPERIMENTAL SETUP: MOSES

Moses is a **statistical machine translation system** that allows us to automatically train translation models for any language pair. All we need is a collection of translated texts (parallel corpus).

- **beam-search**: an efficient search algorithm finds quickly the highest probability translation among the exponential number of choices
- **phrase-based**: the state-of-the-art in statistical machine translation allows the translation of short text chunks
- **factored**: words may have factored representation (surface forms, lemma, part-of-speech, morphology, word classes...)

1 STEP-BY-STEP INSTALLATION

1.1 Get the latest release of Moses

First of all we need to download the latest release of Moses. To do so, we have to install SVN (subversion) which is a version control utility.

To install it, write in a shell:

```
$ sudo apt-get install subversion
```

Then, obtain the latest:

```
$ mkdir ~/mosesdecoder  
$ cd ~  
$ svn co https://svn.sourceforge.net/svnroot/mosesdecoder/trunk mosesdecoder
```

This will copy all of the Moses source code to your local machine.

1.2 Get SRILM

SRILM is a toolkit for building and applying statistical language models (LMs), primarily for use in speech recognition, statistical tagging and segmentation. It has been under development in the SRI Speech Technology and Research Laboratory since 1995. Moses depends on SRILM to compile and to create LM's and translations.

We can download SRILM code from:

<http://www.speech.sri.com/projects/srilm/download.html>.

Decompress it to the location where we want to keep it.

```
$ make srilm
$ cd srilm
$ tar -xzvf srilm.tgz
```

(SRILM expands in the current directory, not in a sub-directory).

READ THE INSTALL FILE - there are a lot of tips in there.

We'll now refer to this location as \$SRILM. In order to compile SRILM we'll be needing:

- A template-capable ANSI-C/C++ compiler, [gcc](#) version 3.4.3 or higher
- [GNU make](#), to control compilation and installation.
- [GNU gawk](#), required for many of the utility scripts.
- [GNU gzip](#) to unpack the distribution and to allow SRILM programs to handle compressed datafiles (highly recommended).
- The [Tcl](#) embeddable scripting language library (only required for some of the test executables)

edit Makefile to point to your directory. Here's my diff:

```
< # SRILM = $SRILM /devel
---
> SRILM = /home/jschroe1/demo/tools/srilm
```

Execute the following single command for all the above:

```
$ sudo apt-get install g++ make gawk gzip tcl8.4 tcl8.4-dev
$ csh
```

1.3 Install SRILM

Note that this package does not come with a configuration script, which makes harder to compile it. The first thing we should modify is our \$SRILM/Makefile.

Write:

```
$ cp $SRILM/Makefile $SRILM/Makefile.bak
$ gedit $SRILM/Makefile
```

READ THE INSTALL FILE - there are a lot of tips in there.

and include these lines at the top of the file:

```
$ SRILM=absolutepathtotheSRILMfolder($SRILM)
$ MACHINE_TYPE=i686 (depends on your machine; check using "$uname -m")
```

We may also need to modify the machine-specific makefile

```
$ cp $SRILM/common/Makefile.machine.i686
$SRILM/common/Makefile.machine.i686.bak
$ gedit $SRILM/common/Makefile.machine.i686
```

Look for CC and replace with the following:

```
CC = /usr/bin/gcc$(GCC_FLAGS)
CXX = /usr/bin/g++$(GCC_FLAGS)-DINSTANTIATE_TEMPLATES
```

Look for TCL_INCLUDE and replace with the following:

```
TCL_INCLUDE = -I/usr/include/tcl8.4/
TCL_LIBRARY = /usr/lib/libtcl8.4.so
```

Now we are ready to compile:

```
$ cd $SRILM
$ sudo make
```

If no errors appeared, then we can proceed with the installation

```
$ sudo make World
```

Now we have to include the \$SRILM/bin/ to the \$PATH environment variable:

```
$ export PATH=$SRILM/bin:$SRILM/bin/i686:$PATH
```

And we are done with SRILM!

1.4 Compile Moses

Now we are ready to compile moses decoder. Beforehand, we need to install some extra packages (autoconf, automake, makeinfo (texinfo), cshzlib)

```
$ sudo apt-get install autoconf automake texinfo zlib1g zlib1g-dev zlib-bin
zlibc
```

Next, we need to regenerate the makefiles. To do so, run the following script:

```
$ cd ~/mosesdecoder
$ ./regenerate-makefiles.sh
```

And configure for compilation:

```
$ cd ~/mosesdecoder
$ ln -s $SRILM ./
$ env LDFLAGS=-static && ./configure --with-srilm=$SRILM
```

and compile:

```
$ cd ~/mosesdecoder
$ make -j 4
```

1.5 Install GIZA++

GIZA++ is an extension of the program GIZA (part of the SMT toolkit [EGYPT](#)) which was developed by the Statistical Machine Translation team during the summer workshop in 1999 at the Center for Language and Speech Processing at Johns-Hopkins University (CLSP/JHU). GIZA++ includes a lot of additional features. The extensions of GIZA++ were designed and written by [Franz Josef Och](#).

First of all, we'll need to get GIZA++

```
$ mkdir ~/GIZA++  
$ cd GIZA++
```

```
$ wget http://www.fjoch.com/GIZA++.2003-09-30.tar.gz  
$ tar xzf GIZA++.2003-09-30.tar.gz
```

Or

```
$ wget http://giza-pp.googlecode.com/files/giza-pp-v1.0.2.tar.gz  
$ tar -xzvf giza-pp-v1.0.2.tar.gz
```

Note that in order to compile GIZA++ we'll need g++-3.3 so, simply install it.

```
$ sudo apt-get install g++-3.3
```

Now go to the GIZA++-v2 folder and modify the Makefile making these changes:

```
CXX = g++-3.3
```

And

```
opt: GIZA++ snt2plain.out plain2snt.out snt2cooc.out
```

```
Save the file and compile GIZA++  
$ make GIZA++
```

1.6 Install mkcls

mkcls is a tool to train word classes by using a maximum-likelihood-criterion. The resulting word classes are especially suited for language models or statistical translation models. The program *mkcls* was written by [Franz Josef Och](#).

Get mkcls:

```
$ mkdir ~/mkcls  
$ cd mkcls  
$ wget http://www.fjoch.com/mkcls.2003-09-30.tar.gz  
$ tar xzf mkcls.2003-09-30.tar.gz
```

Now go to the mkcls-v2 folder and modify the Makefile making these changes to change the compiler

```
directive:  
CFLAGS = -Wall -W -DNDEBUG -O3 -Wno-deprecated  
CXX=g++-3.3
```

Save the file and compile mkcls

```
$ make mkcls
```

1.7 Compile training scripts

Now we need to compile the training scripts. First of all, we need to create an exports dir and place there all the dependencies.

```
$ cd ~/mosesdecoder/scripts
$ mkdir exports
$ cd exports
$ cp ~/GIZA++/GIZA++-v2/GIZA++ ~/GIZA++/GIZA++-v2/snt2cooc.out ./
$ cp ~/mkcls/mkcls-v2/mkcls ./
```

Then, create a release folder and export it as SCRIPTS_ROOTDIR:

```
$ cd ~/mosesdecoder/scripts
$ mkdir release
$ export SCRIPTS_ROOTDIR=~/mosesdecoder/scripts/release
```

Remember to export that variable also to your ~/.bashrc profile. After that, modify the scripts/Makefile so we specify those directories:

```
TARGETDIR=~/mosesdecoder/scripts/release
BINDIR=~/mosesdecoder/scripts/exports
```

Finally compile the scripts.

```
$ cd ~/mosesdecoder/scripts
$ make
```

And we are done !!!

1.8 Compile training scripts

There are few scripts not included with moses which are useful for preparing data

```
$ cd ~
$ wget http://homepages.inf.ed.ac.uk/jschroe1/how-to/scripts.tgz
$ tar -xzvf scripts.tgz
```

We'll also get a NIST scoring tool.

```
$ wget ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl
$ chmod +x mteval-v11b.pl
```

2 TRAINING AND TESTING

We have now a Moses decoder that will translate for us some input texts by using **phrase tables**. Nevertheless, in order to have good phrase tables, we need to train the engine with some corpora. Here is how we'll do it:

You can use any pair-aligned corpora you may get in hand.

2.1 Prepare Data

2.1.1 Tokenize training data

```
$ mkdir corpus
```

Download the parallel corpus in this directory as corpus.hi and corpus.en

```
$ cat corpus.hi | $SCRIPTS_DIR/tokenizer.perl -l hi > corpus/corpus.tok.hi
$ cat corpus.en | $SCRIPTS_DIR/tokenizer.perl -l en > corpus/corpus.tok.en
```

2.1.2 Filter out long sentences

```
$ $SCRIPTS-YYYYMMDD-HHMM-DIR/training/clean-corpus-n.perl corpus/corpus.tok
fr en corpus/corpus.clean 1 40
```

2.1.3 Lowercase training data

```
$ $SCRIPTS_DIR /lowercase.perl < corpus/corpus.clean.hi >
corpus/corpus.lowercased.hi
$ $SCRIPTS_DIR /lowercase.perl < corpus/corpus.clean.en >
corpus/corpus.lowercased.en
```

(Do not lowercase Hindi corpus in case lower and upper case denote different pronunciations)

2.2 Build Language Model

Language models are concerned only with n-grams in the data, so sentence length doesn't impact training times as it does in GIZA++. So, we'll lowercase the full 55,030 tokenized sentences to use for language modeling. Many people incorporate extra target language monolingual data into their language models.

```
$ mkdir lm
$ $SCRIPTS_DIR /lowercase.perl < corpus/ corpus.tok.en > lm/
corpus.lowercased.en
```

We will use SRILM to build a tri-gram language model.

```
$ $SRILM-HOME/bin/i686/ngram-count -order 3 -interpolate -kndiscount -unk -
text lm/corpus.lowercased.en -lm lm/corpus.lm
```

2.3 Training

Moses' toolkit does a great job of wrapping up calls to mkcls and GIZA++ inside a training script, and outputting the phrase and reordering tables needed for decoding. The script that does this is called `train-factored-phrase-model.perl`

We'll run this in the background and nice it since it'll peg the CPU while it runs. It may take up to an hour, so this might be a good time to run through the tutorial page mentioned earlier using the sample-models data.

```
$ nohup nice $SCRIPTS-YYYYMMDD-HHMM/training/train-factored-phrase-model.perl
-scripts-root-dir $SCRIPTS-YYYYMMDD-HHMM -root-dir work -corpus
corpus/corpus.lowercased -f en -e hi -alignment grow-diag-final-and -
reordering msd-bidirectional-fe -lm
0:3:ABSOLUTE_PATH_TO_CURRENT_DIR/lm/corpus.lm >& work/training.out &
```

You can

```
$ tail -f work/training.out
```

file to watch the progress of the tuning script. The last step will say something like:

```
(9) create moses.ini @ Tue Jan 27 19:40:46 CET 2009
```

2.4 Tuning

Download Tuning sets (same corpus used earlier may be used) in the work directory as `tuning-set.en`. Also get the test corpus into evaluation directory – `test.hi` and `test.en`

```
$ nohup nice $SCRIPTS-YYYYMMDD-HHMM-DIR/training/mert-moses.pl work/tuning-
set.en work/tuning/tuning-set.en $MOSES_HOME/moses-cmd/src/moses
work/model/moses.ini --working-dir work/tuning/mert --rootdir $SCRIPTS-
YYYYMMDD-HHMM-DIR/ --decoder-flags "-v 0" >& work/tuning/mert.out &
```

2.5 Generating output

```
$ mkdir evaluation
$ $MOSES_HOME/moses-cmd/src/moses -config tuning/moses.ini -input-file
test.en > evaluation/test.output;
```

2.6 Adding tags

(evaluation folder contains the test corpus – test.hi and test.en)

```
$ awk -f addtag_tst.awk evaluation/test.output > out
$ awk -f addtag_ref.awk evaluation/test.hi > ref
$ awk -f addtag_src.awk evaluation/test.en > src
```

2.4 Bleu Score Calculations

```
$ MOSES_HOME/scripts/mteval-v11b.pl -r ref -t out -s src -c
```

3 MISCELLANEOUS

3.1 Script- All in one

(For English-Hindi MT)

Make necessary changes to point to your installation and corpus directories.

```
#!/bin/bash
#echo " ----- cleaning (and lowercase the english only)-----";
../moses/scripts/release/scripts-20090111-1339/training/clean-corpus-n.perl train hi en train.clean 1 50;

#echo " ----- Building language model -----";
mkdir lm;
../srilm/bin/i686-m64/ngram-count -order 3 -interpolate -kndiscount -text train.surface.hi -lm
lm/surf.lm;

#echo " ----- Training model -----";
../moses/scripts/release/scripts-20090111-1339/training/train-factored-phrase-model.perl --scripts-
root-dir ../moses/scripts/release/scripts-20090111-1339 --root-dir . --corpus train.clean --e hi --f en --lm
0:3:/home/hansraj/ddp_exp/surfaceLR/lm/surf.lm:0 -reordering distance,msd-bidirectional-fe;

#echo " ----- Tuning -----"
mkdir tuning
cp tun.en tuning/input
cp tun.hi tuning/reference
/home/hansraj/ddp_exp/moses/scripts/release/scripts-20090111-1339/training/mert-moses.pl
```

```
tuning/input tuning/reference /home/hansraj/ddp_exp/moses/moses-cmd/src/moses
/home/hansraj/ddp_exp/surfaceLR/model/moses.ini --working-dir
/home/hansraj/ddp_exp/surfaceLR/tuning --rootdir
/home/hansraj/ddp_exp/moses/scripts/release/scripts-20090111-1339
```

```
#echo " ----- Generating output -----";
mkdir evaluation;
../moses/moses-cmd/src/moses -config tuning/moses.ini -input-file test.en >evaluation/test.output;
```

```
echo " ----- Adding tags -----";
awk -f addtag_tst.awk evaluation/test.output >out;
awk -f addtag_ref.awk test.hi >ref;
awk -f addtag_src.awk test.en >src;
```

```
echo " ----- Bleu Score Calculations -----";
../moses/scripts/mteval-v11b.pl -r ref -t out -s src -c
```
