**STEP BY STEP GUIDE FOR ENGLISH TO HINDI TRANSLATION USING MOSES**

IMPORTANT: Preferably install all the given tools and data in the home directory (i.e. ~/).

**STEP 1: Downloading Moses and Installing Necessary Packages**

1. Open Terminal.
2. git clone <https://github.com/moses-smt/mosesdecoder.git>
3. sudo apt-get install [package name]

PACKAGES: 1. g++

2. git

3. subversion

4. automake

5. libtool

6. zlib1g-dev

7. libboost-all-dev

8. libbz2-dev

9. liblzma-dev

10. python-dev

11. graphviz

12. imagemagick

13. make

14. cmake

15. libgoogle-perftools-dev

16. autoconf

17. doxygen

**NOTE**: In case you get the error “Unable to locate package”, try re-generating your /etc/apt/sources.list from the given link : [https://repogen.simplylinux.ch](https://repogen.simplylinux.ch/).

**STEP 2: Boost Installation**

1. wget <https://dl.bintray.com/boostorg/release/1.64.0/source/boost_1_64_0.tar.gz>
2. tar zxvf boost\_1\_64\_0.tar.gz
3. cd boost\_1\_64\_0/
4. sudo apt-get update
5. sudo apt-get install build-essential g++ python-dev autotools-dev libicu-dev build-essential libbz2-dev
6. ./bootstrap.sh --prefix=/usr/local
7. n=`cat /proc/cpuinfo | grep "cpu cores" | uniq | awk '{print $NF}'`
8. sudo ./b2 --with=all -j $n install
9. sudo sh -c 'echo "/usr/local/lib" >> /etc/ld.so.conf.d/local.conf'
10. sudo ldconfig

**STEP 3: Installing CMPH**

1. Download cmph-2.0 from: <https://sourceforge.net/projects/cmph/>
2. tar zxvf cmph-2.0.tar.gz

**STEP 4: Installing Moses**

NOTE: This is the most important step and usually takes a lot of time.

1. cd mosesdecoder
2. ./bjam --with-boost=~/boost\_1\_64\_0 -j8

**NOTE**: If you do not get the “SUCCESS” message at the end try the following command:

./bjam -a –with-boost=~/boost\_1\_64\_0 --with-cmph=~/cmph-2.0 -j8

If the problem still exists try adding –no-xmlrpc-c to your bjam command.

**STEP 5: Installing Mgiza (in the home directory)**

1. sudo apt-get install cmake
2. git clone <https://github.com/moses-smt/mgiza.git>
3. cd mgiza/mgizapp
4. cmake .
5. make
6. make install

**STEP 6: Copying necessary files**

1. mkdir workspace
2. cd workspace
3. mkdir bin
4. cd bin
5. mkdir training-tools
6. cd training-tools
7. mkdir mgizapp
8. cd ~/mgiza/mgizapp
9. export BINDIR=~/workspace/bin/training-tools
10. cp bin/\* $BINDIR/mgizapp
11. cp scripts/merge\_alignment.py $BINDIR
12. cd ~/mosesdecoder
13. mkdir tools
14. Copy merge\_aligment.py from ~/workspace/bin/training\_tools to ~/mosesdecoder/tools
15. Copy mgiza, mkcls, snt2cooc from ~/workspace/bin/training\_tools/mgizapp to ~/mosesdecoder/tools

**STEP 7: Installing IRSTLM**

1. wget <http://sourceforge.net/projects/irstlm/files/irstlm/irstlm-5.80/irstlm-5.80.06.tgz/download>
2. tar zxvf irstlm-5.80.06.tgz
3. cd irstlm-5.80.08/trunk
4. ./regenerate-makefiles.sh
5. ./configure --prefix=$HOME/irstlm
6. make install

**STEP 8: For Indian Languages**

For Natural Language Processing in Indian languages, install the following library:

Either git clone this repository or <https://github.com/anoopkunchukuttan/indic_nlp_library>

NOTE: Make sure that the “indic\_nlp\_library-master” folder is in the home directory.

**STEP 9: Downloading Corpus**

1. Make a new directory “corpus” in the home directory. Make a directory “training” inside the directory corpus.
2. Download  [corpora.en](https://drive.google.com/file/d/0B7mGVKJlRRlueGhHeDRucnRfMVU/view?usp=sharing" \t "_blank) and [corpora.hi](https://drive.google.com/open?id=0B7mGVKJlRRluWEFXRVN4MFJtNVk" \t "_blank). Put these two files in the training directory.

**The home directory now contains the following necessary folders: mosesdecoder, corpus, indic\_nlp\_library-master, irstlm-5.80.08, irstlm, boost\_1\_64\_0, cmph-2.0, mgiza and workspace.**

**STEP 10: Pre-process corpora**

To prepare the data for training the translation system, we have to perform the following steps:

* **tokenisation**: This means that spaces have to be inserted between (e.g.) words and punctuation.
* **truecasing**: The initial words in each sentence are converted to their most probable casing. This helps reduce data sparsity.
* **cleaning**: Long sentences and empty sentences are removed as they can cause problems with the training pipeline, and obviously mis-aligned sentences are removed.

**HINDI PREPARATION**

1. Normalization:

cd corpus

python ~/indic\_nlp\_library-master/src/indicnlp/normalize/indic\_normalize.py training/corpora.hi corpora.norm.hi [True]

1. Tokenization:

python ~/indic\_nlp\_library-master/src/indicnlp/tokenize/indic\_tokenize.py corpora.norm.hi corpora.tok.hi hi

1. Creating truecase model:

~/mosesdecoder/scripts/recaser/train-truecaser.perl –model truecase-model.hi –corpus corpora.tok.hi

1. Truecasing:

~/mosesdecoder/scripts/recaser/truecase.perl –model truecase-model.hi < corpora.tok.hi > corpora.true.hi

**ENGLISH PREPARATION**

1. Tokenization:

~/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en < training/corpora.en > corpora.tok.en

1. Creating truecase model:

~/mosesdecoder/scripts/recaser/train-truecaser.perl –model truecase-model.en –corpus corpora.tok.en

1. Truecasing:

~/mosesdecoder/scripts/recaser/truecase.perl –model truecase-model.en < corpora.tok.en > corpora.true.en

**CLEANING OF ENGLISH AND HINDI**

~/mosesdecoder/scripts/training/clean-corpus-n.perl corpora.true en hi corpora.clean 1 80

**STEP 11: Language Model Buiding**

* 1. cd ~
  2. mkdir lm
  3. cd lm
  4. ~/irstlm/bin/add-start-end.sh < ~/corpus/corpora.clean.hi > corpora.sb.hi
  5. export IRSTLM=/home/administrator/irstlm/
  6. ~/irstlm/bin/build-lm.sh -i corpora.sb.hi -t ./tmp -p -s improved-kneser-ney -o corpora.lm.hi
  7. ~/irstlm/bin/compile-lm corpora.lm.hi.gz –text=yes corpora.arpa.hi
  8. ~/mosesdecoder/bin/build\_binary -i corpora.arpa.hi corpora.blm.hi

**STEP 12: Training**

1. cd ~
2. mkdir working
3. cd working
4. nohup nice ~/mosesdecoder/scripts/training/train-model.perl -mgiza -root-dir train --corpus ~/corpus/corpora.clean -f en -e hi -alignment grow-diag-final-and -reordering msd-bidirectional-fe -lm 0:3:/home/administrator/lm/corpora.blm.hi:8 -external-bin-dir ~/mosesdecoder/tools

>& training.out &