Moses User Guide

(CSLT-TRP-160018)

冯洋 (Yang Feng)
yangfeng145@gmail.com

CSLT, RIIT, Tsinghua Univ.
NLP Group
2016-12-15

目录

1. 介绍

Moses 是用于机器翻译的工具，其他所有的从源端序列生成目标端序列的问题也可以看成是机器翻译问题，比如问题改写，从而使用 Moses 进行建模。下面笔者着重介绍两个模型：短语模型和层次短语模型。

短语模型主要是通过查找与源端匹配的短语，进而将短语进行拼接从而生成目标译文。比如，对于一个等价的问题对

```
什么 是 成人 高考 ||| 成人 高考 介绍
```

短语模型会抽取以下短语

```
什么 是 → 介绍
成人 高考 → 成人 高考
```

这样，如果给定『什么 是 成人 高考』，通过应用以上两条规则，就可以得到『成人 高考 介绍』。短语模型存在一个问题是，它并不知道『成人 高考』和『介绍』的位置关系，只能靠语言模型来对排序进行弱指导，从而使得生成正确的句子排序有一定的挑战。

相比于短语模型，层次短语模型引入了占位符 X，从而能学习一些特殊的句型变换。对于以上的问题对， 层次短语就能学到以下规则

```
什么 是 X → X 介绍
成人 高考 → 成人 高考
```

这样，如果给定『什么 是 成人 高考』，通过应用以上两条规则，就可以得到『成人 高考 介绍』

2. 编译

将 Moses 文件夹存放在某个目录下，进入 Moses 目录，在命令行输入以下命令：

```
export MOSES=$PWD
```

2.1 依赖

2.1.1 gcc

请保证 gcc 的版本不要太低，最好是 gcc4.8

2.1.2 boost

boost 版本需要至少是 1.48，我用的是 1.62

export $BOOST=$MOSES/usr/boost

```
./bootstrap.sh
./b2 --prefix=$BOOST --libdir=$BOOST/lib64 --layout=tagged
link=static,shared threading=multi install
```

2.1.3 bzip2

需安装 bzip2，请参考 http://www.bzip.org/

修改~/.bashrc 的 PATH, LD_LIBRARY_PATH，以包括以上工具的 bin 和对应的 lib 或者 lib64 目录。

2.2 SRILM

进入$MOSES/srilm 目录。

首先修改 Makefile 文件的 SRILM 的路径，改成为现在 srilm 的根目录，为

```
SRILM = $MOSES/srilm
```

最后运行命令

```
make World
```

至此，srilm 编译完成，检查$MOSES/srilm/bin/i686-m64/ ngram-count 是否可以运行，如果可以运行，证明编译成功。

2.3 GIZA++

进入$MOSES/giza-pp 目录，运行 make 直接进行编译，检查 GIZA++-v2 目录下是否有以下可执行文件：

```
GIZA++, plain2snt.out, snt2cooc.out, snt2plain.out
```

将这些可执行文件拷贝到 giza-pp 目录下。

2.4 Moses

进入$MOSES/mosesdecoder 目录，运行以下命令进行编译：

```
./bjam -aq -j 4 --with-boost=$BOOST--with-srilm=$MOSES/srilm
```

如果编译成功，在**$MOSES/mosesdecoder/bin** 目录下有 **moses** 和 **moses_chart** 可执行程序，如果运行会输出相应的命令选项。

3. 运行短语模型

3.1 数据准备

创建以下目录

```
$MOSES/workplace,
$MOSES/workplace/data,
$MOSES/workplace/dev_test
```

data 目录用来存放训练集。源句子和目标句子分别一个文件，一个句子一行。

dev_test 目录用来存放开发集、测试集。开发集和测试集都是一个句子一行，不加任何标记，对于参考译文，一个参考译文一个文件，后缀从 0 开始算起。这里开发集选定 dev，测试集选定 test，所以用到的文件为 dev.src，dev.ref.0，dev.ref.1, dev.ref.2, dev.ref.3, test.src, test.ref.0, test.ref.1, test.ref.2, test.ref.3。

3.2 训练语言模型

创建语言模型目录并进入该目录

```
mkdir $MOSES/lm
```

用 srilm 训练语言模型，运行以下命令或者./train_demo.sh

```
../srilm/bin/i686-m64/ngram-count -text demo_lm_plain.txt -order 5
-unk -interpolate -wbdiscount -lm demo.o5.lm.gz
```

3.3 抽取短语

进入$MOSES/workplace_demo 目录，运行**./train.sh** 或者以下命令

```
$MOSES/mosesdecoder/scripts/training/train-model.perl \
    --external-bin-dir $MOSES/giza-pp \
    --root-dir $MOSES/workplace_demo/train \
    --corpus $MOSES/workplace_demo/data/training \
    -f source \
    -e target \
    --first-step 1 \
    --last-step 9 \
    --alignment grow-diag-final-and \
    --reordering msd-backward-fe \
    --lm 0:5:$MOSES/lm/demo.o5.lm.gz
```

其中，每个参数的含义如下：

--external-bin-dir: 为引入的外部程序库的bin目录，此处为GIZA++对齐的目录

--root-dir: 为生成的文件的存放目录

--corpus: 为训练文件的前缀

-f source: 为源端的训练文件的后缀

-e target: 为目标端的训练文件的后缀

--first-step: 为开始的步骤

--last-step: 为结束的步骤

--alignment: 为生成的对齐的种类

--reordering: 为调序模型的种类

--lm: 为使用的语言模型，有三个参数，用冒号隔开，为语言模型种类：语言模型元数：语言模型文件，srilm 对应的种类为 0.

运行完以后会自动生成$MOSES/workplace_demo/train 目录，如果运行成功，在$MOSES/workplace_demo/train/model 目录下会有以下三个文件供 mert 训练阶段使用：

phrase-table.gz (短语表)

reordering-table.wbe-msd-backward-fe.gz (排序模型文件)

moses.ini (配置文件)

其中，moses.ini 文件中会给出训练阶段的参数值。

3.4 mert 训练

抽取完短语表之后，就可以进行 mert 训练来训练参数。运行**./mert.sh** 或者输入以下命令：

```
$MOSES/mosesdecoder/scripts/training/mert-moses.pl \
    --working-dir $MOSES/workplace_demo/mert \
    --rootdir $MOSES/mosesdecoder/scripts\
    --decoder $MOSES/mosesdecoder/bin/moses \
    --input $MOSES/workplace_demo/dev_test/dev.src\
    --refs $MOSES/workplace_demo/dev_test/dev.ref. \
    --config $MOSES/workplace_demo/train/model/moses.ini \
    --nbest 100 --jobs 4 \
    --return-best-dev \
    --filter-phrase-table \
    --queue-flags "-q wolf.cpu.q"
```

其中每个参数的含义如下：

--working-dir:　存放生成文件的目录

--rootdir: 使用的脚本文件的目录

--decoder: 使用的解码器可执行文件

--input: 开发集的源文件

--refs: 开发集的参考文件的前缀，系统会自动添加数字0，1，……

--config: 配置文件

--nbest: nbest size

--jobs: 并发任务数

--return-best-dev:　表示返回对应于bleu值最高的轮数的参数

--filter-phrase-table: 表示对短语进行过滤，只保留与开发集源文件匹配的短语

--queue-flags: 与机器相关，设置成对应于机器的队列名

如果中断了，要继续进行 mert 训练，只需要添加选项—continue. 训练的时候会默认的对短语表和重排序表进行过滤，实际用到的是 filetered 目录下的 moses.ini，phrase-table.0-0.1.1.gz 和 reordering-table.wbe-msd-backward-fe。

训练结束后，得到的$MOSES/workplace_demo/mert/moses.ini 便为训练得到的配置文件。

3.5 测试

首先创建目录 $MOSES/workplace_demo/test, 然后将 $MOSES/workplace_demo/mert/moses.ini 拷贝到新创建的目录下。对

于大的训练语料，必须对测试集进行短语表的过滤，小的训练语料也可以不进行过滤。

```
$MOSES/mosesdecoder/scripts/training/filter-model-given-input.pl \
    ./test/filtered \
    ./test/moses.ini \
    ./dev_test/test.src
```

三个参数分别为：过滤后得到的短语表以及配置文件的存放目录，测试使用的配置文件，测试文件。

过滤后的配置文件 moses.ini，短语表以及重排序表存放在目录$MOSES/workplace_demo/test/filtered。

测试运行以下命令

```
$MOSES/mosesdecoder/bin/moses  \
    -config ./test/filtered/moses.ini \
    -n-best-list ./test/nbest.txt 100 \
    < ./dev_test/test.src > ./test/result.txt
```

其中参数的含义为

```
-config: 测试使用的配置文件
-n-best-list: 生成的nbest文件以及nbest_size
< 测试文件
> 结果文件，一个测试句子对应一个最好的结果
```

生成的结果文件 result.txt 是一个 plain 文件，一行一个结果。如果需

要测试最终结果的 BLEU 值，还需要运行

```
$MOSES/mosesdecoder/scripts/generic/multi-bleu.perl  ./dev_test/test.ref. < ./test/result.txt > ./test/result.eval
```

其中对应的参数为

```
第一个参数是参考文件，只需要给出前缀，程序会自动加上数字来寻找，即程序会查找 test.ref.0, test.ref.1, …
< 要进行评估的结果文件
> 存放评估结果的文件
```

整个测试过程均写在脚本 **test.sh**，可直接运行。

4. 运行层次短语模型

4.1 抽取规则

运行以下命令或者直接运行**./train_hpb.sh**

```
$MOSES/mosesdecoder/scripts/training/train-model.perl \
    --external-bin-dir $MOSES/giza-pp \
    --root-dir $MOSES/workplace_demo/train_hpb \
    --corpus $MOSES/workplace_demo/data/training \
    -f source \
    -e target \
    --first-step 1 \
    --last-step 9 \
    --alignment grow-diag-final-and \
    --lm 0:5:$MOSES/lm/demo.o5.lm.gz \
    --hierarchical \
    --glue-grammar \
    --extract-options '--MinHoleSource 1' \
    --extract-options '--NoNonTermFirstWord' \
    --score-options '--GoodTuring'
```

其中每个参数对应的含义为

--external-bin-dir: 为引入的外部程序库的bin目录，此处为GIZA++对齐的目录

--root-dir: 为生成的文件的存放目录

--corpus: 为训练文件的前缀

-f source: 为源端的训练文件的后缀

-e target: 为目标端的训练文件的后缀

--first-step: 为开始的步骤

--last-step: 为结束的步骤

--alignment: 为生成的对齐的种类

--reordering: 为调序模型的种类

--lm: 为使用的语言模型，有三个参数，用冒号隔开，为语言模型种类：语言模型元数：语言模型文件，srilm 对应的种类为 0.

--hierarchical：表示抽取层次短语规则

--glue-grammar：层次短语模型所需要的glue规则

--extract-options: 抽取规则的附加参数

--score-options: 给规则评分所用到的参数

运行成功，会在$MOSES/workplace_demo/train_hpb/model 生成三个文件以供 mert 训练使用:

phrase-table.gz

rule-table.gz

moses.ini

4.2 mert 训练

运行以下命令或者直接运行**./mert_hpb.sh**

```
$MOSES/mosesdecoder/scripts/training/mert-moses.pl \
    --working-dir $MOSES/workplace_demo/mert_hpb \
    --rootdir $MOSES/mosesdecoder/scripts\
    --decoder $MOSES/mosesdecoder/bin/moses_chart \
    --input $MOSES/workplace_demo/dev_test/dev.src\
    --refs $MOSES/workplace_demo/dev_test/dev.ref. \
    --config $MOSES/workplace_demo/train_hpb/model/moses.ini \
    --nbest 100 --jobs 4 \
    --return-best-dev \
    --filter-phrase-table \
    --queue-flags "-q wolf.cpu.q"
```

其中每个参数的含义如下：

--working-dir:　存放生成文件的目录

--rootdir: 使用的脚本文件的目录

--decoder: 使用的解码器可执行文件

--input: 开发集的源文件

--refs: 开发集的参考文件的前缀，系统会自动添加数字0，1，……

--config: 配置文件

--nbest: nbest size

--jobs: 并发任务数

--return-best-dev:　表示返回对应于bleu值最高的轮数的参数

--filter-phrase-table: 表示对短语进行过滤，只保留与开发集源文件匹配的短语

--queue-flags: 与机器相关，设置成对应于机器的队列名

训练好的参数为$MOSES/workplace_demo/mert_hpb/moses.ini

4.3 测试

首先创建目录$MOSES/workplace_demo/test_hpb, 然后将$MOSES/workplace_demo/mert_hpb/moses.ini 拷贝到新创建的目录下。此处不进行规则表的过滤。

测试运行以下命令

```
$MOSES/mosesdecoder/bin/moses_chart    \
    -config ./test_hpb/moses.ini \
    -n-best-list ./test_hpb/nbest.txt 100 \
    < ./dev_test/test.src > ./test_hpb/result.txt
```

其中参数的含义为

```
-config: 测试使用的配置文件
-n-best-list: 生成的nbest文件以及nbest_size
< 测试文件
> 结果文件，一个测试句子对应一个最好的结果
```

生成的结果文件 result.txt 是一个 plain 文件，一行一个结果。如果需要测试最终结果的 BLEU 值，还需要运行

```
$MOSES/mosesdecoder/scripts/generic/multi-bleu.perl  ./dev_test/test.ref. < ./test_hpb/result.txt > ./test_hpb/result.eval
```

其中对应的参数为

```
第一个参数是参考文件，只需要给出前缀，程序会自动加上数字来寻找，即程序会查找 test.ref.0, test.ref.1, …
< 要进行评估的结果文件
> 存放评估结果的文件
```

整个测试过程均写在脚本 **test_hpb.sh**，可直接运行。