# Centos7 中安装moses

**源码地址**：git clone <https://github.com/moses-smt/mosesdecoder.git>

**注意权限问题**

**系统不被环境：**

git

subversion

make

libtool

gcc

g++

libboost-dev

tcl-dev

tk-dev

zlib1g-dev

libbz2-dev

python-dev

**sudo yum install git subversion make libtool gcc g++ libboost-dev tcl-dev tk-dev zliblg-dex libbz2-dev python-dev**

 G++没有装或却没有更新

以下方法都可以试试：

centos：

yum -y update gcc

yum -y install gcc+ gcc-c++

**需要工具:**

下载地址: http://downloads.sourceforge.net/project

1. boost 地址：<http://sourceforge.net/projects/boost/files/boost/1.63.0/boost_1_63_0.tar.gz/download>

[安装方式](#_Boost安装方式)

1. srilm 地址：

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

[安装方式](#_srilm安装方式)

1. irstlm地址： https://managedway.dl.sourceforge.net/project/irstlm/irstlm/irstlm-5.80/irstlm-5.80.08.zip

[安装方式](#_irstlm安装方式)

1. 分词工具: 斯坦福

地址: <https://nlp.stanford.edu/software/stanford-segmenter-2016-10-31.zip>

安装方式解压即可

1. Cmph 地址<https://svwh.dl.sourceforge.net/project/cmph/cmph/cmph-2.0.tar.gz>

[安装方式](#_cmph安装方式)

1. Xmlrpc 地址<https://sourceforge.net/projects/xmlrpc-c/files/Xmlrpc-c%20Super%20Stable/1.39.12/xmlrpc-c-1.39.12.tgz/download>

安装方式

**语料准备与处理:**

1)：准备语料库：联合国语料库(https://conferences.unite.un.org/UNCorpus/)

MGIZA: <https://github.com/moses-smt/mgiza.git>[(安装)](#_mgiza安装方式)

[语料库处理](#_语料库处理)

[Language Model Training（语言模型处理）](#_Language_Model_Training（语言模型处理）)

[Training the Translation System](#_Training_the_Translation)

[Moses server 配置](#_Moses_server_配置)

[ems](#_EMS系统)

[lamp 安装](#_lampp_安装)

[参数(训练，使用参数)](#_参数)

[Training Syntax Models训练语法模型](#_Training_Syntax_Models)

组件安装：

创建目录 opt 或者其它

手动编译

1 进入moses

2 语言模型工具

1） rstlm 语言模型工具编译

./bjam –j4(数字代表着cpu核心数)

--with-boost=/home/liuyan/new\_moses/opt

--with-cmph=/home/liuyan/new\_moses/opt

--with-xmlrpc-c=/home/liuyan/new\_moses/opt

--with-srilm=/home/liuyan/new\_moses/srilm-1.7.2 > /路径

--with-mm

**./jam-files/bjam -j3 --with-boost=/home/xiaosun/mouse/compile/opt --with-cmph=/home/mouse/compile/opt --with-xmlrpc-c=/home/xiaosun/mouse/compile/opt --with-srilm=/home/xiaosun/compile/srilm --debug-configuration -d2 |gzip >build.log.gz**

# Boost安装方式

1 下载 并且**tar -zxvf boost\_1\_60\_0.tar.gz**

**2 cd boost\_1\_60\_0**

**3./bootstrap.sh –**prefix=安装路径(/home/xiaosun/compile/opt)

4 以管理员执行 ./b2 install 或者**./bjam install**安装

1/etc/ld.so.conf.d/

2创建conf文件

(输入路径：/home/xiaosun/mouse/compile/opt/lib/)

3sudo /sbin/ldconfig -v

新增文件

# cmph安装方式

1 下载 并且**tar -zxvf cmph-2.0.tar.gz**

**2 cd cmph-2.0**

**3**./configure --prefix=/home/xiaosun/mouse/compile/opt

4以管理员执行 make 和 **make install**安装

多了 bin 和share

# srilm安装方式

**需要工具：http://www.tcl.tk/software/tcltk/download.html**

1 下载 并且**tar -zxvf srilm-1.7.2.tar.gz(这里还有一个srilm noiconv)**

**2 cd srilm-1.7.2**

**3 修改Makefile**

**1)找到：**



**添加一行**

**SRILM =你当前解压srilm后的文件目录(/home/xiaosun/mouse/compile/srilm-1.7.2)**

**2)** 注释修改为

方法一:有注释则去掉没有则不管

方法二: MACHINE\_TYPE := i686-m64

3) 创建srilm-1.7.2/common/Makefile.site.i686-m64 如果不需要iconv 则需要添加NO\_ICONV=1

相对而言如果使用**noiconv则需要改成SYS\_LIBRARIES=-L/usr/local/lib -liconv**

4）编译

Make

make install

make World  注意大写

优化

gnumake World OPTION=\_c  
     gnumake cleanest OPTION=\_c

测试：make test

# 斯坦福

命令行:

java -mx3g -cp "\*" edu.stanford.nlp.pipeline.StanfordCoreNLP -props StanfordCoreNLP-chinese.properties -file chinese.txt -outputFormat text

java -mx1g -cp "\*" edu.stanford.nlp.pipeline.StanfordCoreNLP -props StanfordCoreNLP-french.properties -annotators tokenize,ssplit,pos,depparse -file french.txt -outputFormat conllu

多个文件

java -cp "\*" -Xmx2g edu.stanford.nlp.pipeline.StanfordCoreNLP **[** -props myprops.props **]** -filelist filelist.txt

使用内置脚本

./corenlp.sh -annotators tokenize,ssplit,pos -file wikipedia.txt -outputFormat conll

参数说明

-props :配置文件

-file :输入文件(单个文件)

-filelist :多个文件输入

-mx3g：指定内存

-cp:java指定jar文件依赖

-props :（可选参数）默认情况下，斯坦福corenlp将搜索您的类路径stanfordcorenlp.properties和使用默认包含在分布。

-annotators: (可选参数) tokenization and sentence splitting, POS tagging, lemmatization, NER, parsing, and coreference resolution--（分词参数）

-outputDirectory：指定输出目录，默认当前目录

-outputExtension : Output filenames are the same as input filenames but with -outputExtension added to them (.xml by default).输出名字与输入一致只是输出加上了.xml

-outputFormat:格式化

* “text”: An ad hoc human-readable text format. Tokens, s-expression parse trees, relation(head, dep) dependencies. Output file extension is .out. This is the default output format only if the XMLOutputter is unavailable.
* “xml”: An XML format with accompanying XSLT stylesheet, which allows web browser rendering. Output file extension is .xml. This is the default output format, unless the XMLOutputter is unavailable.
* “json”: JSON. Output file extension is .json. ‘Nuf said.
* “conll”: A tab-separated values (TSV) format. Output extension is .conll. This representation may give only a partial view of an Annotation and doesn’t correspond to any particular CoNLL format. Columns are: wordIndex, token, lemma, POS, NER, head, depRel.
* “conllu”: [CoNLL-U](https://universaldependencies.github.io/docs/format.html) output format, another tab-separated values (TSV) format. Output extension is .conllu. This representation may give only a partial view of an Annotation.
* “serialized”: Produces some serialized version of each Annotation. May or may not be lossy. What you actually get depends on the outputSerializer property, which you should also set. The default is the GenericAnnotationSerializer, which uses the built-in Java object serialization and writes a file with extension .ser.gz.

# irstlm安装方式

**./regenerate-makefiles.sh**

**step 1: ./configure [--prefix=** **/home/xiaosun/mouse/compile/opt/] ...**

**# run "configure --help" to get more details on the compilation options**

**step 2: make**

**step 3: make install**

# xmlrpc-c安装方式

$ ./configure **--prefix=** **/home/xiaosun/mouse/compile/opt/**

$ make

$ make install

# mgiza安装方式

git clone <https://github.com/moses-smt/mgiza.git>

cd mgiza/mgizapp

cmake .

make

make install

另外创建training-tools 目录名字随意（$BINDIR）

进入 mgiza/mgizapp

复制

cp bin/\* $BINDIR/mgizapp

cp scripts/merge\_alignment.py $BINDIR

cmake .   
make   
make install   
在moses中配置mgiza++   
cd ~/working/mosesdecoder/   
mkdir word\_align\_tools   
cp ~/working/mgiza/mgizapp/bin/\* word\_align\_tools/   
cp ~/working/mgiza/mgizapp/scripts/merge\_alignment.py word\_align\_tools/

使用方法：训练的时候使用

 train-model.perl -mgiza-cpus NUMBER

# 语料库处理

* **tokenisation**: This means that spaces have to be inserted between (e.g.) words and punctuation.

**(分词)**

这一步主要是在单词和单词之间或者单词和标点之间插入空白，以便于后续识别和其他操作

* **truecasing**: The initial words in each sentence are converted to their most probable casing. This helps reduce data sparsity.

**（语料的大小写处理，中文没必要大小写，英文,法语…这些语言需要大小写处理）**

初始每句话的字和词组都被转换为没有格式的形式(例如统一为小写）。这有助于减少数据稀疏性问题。

* **cleaning**: Long sentences and empty sentences are removed as they can cause problems with the training pipeline, and obviously mis-aligned sentences are removed.

长句和空语句可引起训练过程中的问题，因此将其删除，同时删除显不对齐句子删除。

**（调整长度一般为80）**

**Tokenization:**

**~/mouse/moses1/scripts/tokenizer/tokenizer.perl -l en < ~/mouse/compile/corpus/en-zh/TED2013.en-zh.en > ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.en**

~/mouse/moses1/scripts/tokenizer/tokenizer.perl -l en < ~/mouse/compile/corpus/en-zh/TED2013.en-zh.zh > ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.zh

还有一个：

./scripts/tokenizer/tokenizer.perl -a -l en | scripts/generic/ph\_numbers.perl –c < (输入源路径) > 输出源TOK-DATA

斯坦福分词使用：

123

Truecaser 提取文本信息

~/mouse/moses1/scripts/recaser/train-truecaser.perl --model ~/mouse/compile/corpus/en-zh/truecaser-model.en --corpus ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.en

~/mouse/moses1/scripts/recaser/train-truecaser.perl --model ~/mouse/compile/corpus/en-zh/truecaser-model.zh --corpus ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.zh

Truecasing

~/mouse/moses1/scripts/recaser/truecase.perl --model ~/mouse/compile/corpus/en-zh/truecaser-model.en < ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.en > ~/mouse/compile/corpus/en-zh/TED2013.en-zh.true.en

**Cleaning**

~/mosesdecoder/scripts/training/clean-corpus-n.perl \

~/corpus/news-commentary-v8.fr-en.true fr en \

~/corpus/news-commentary-v8.fr-en.clean 1 80

/home/xiaosun/mouse/md2/scripts/training/clean-corpus-n.perl /home/xiaosun/mouse/compile/corpus/en-zh/TED2013.en-zh.true en zh /home/xiaosun/mouse/compile/corpus/en-zh/TED2013.en-zh.clean 1 80

# Language Model Training（语言模型处理）

1/etc/ld.so.conf.d/

2创建conf文件

(输入路径：/home/xiaosun/mouse/compile/opt/lib/)

3sudo /sbin/ldconfig -v

只能是目标语言:例如en到zh值目标语言是zh（错误5）

~/mouse/md2/bin/lmplz -o 3 < ~/mouse/compile/corpus/en-zh/TED2013.en-zh.tok.zh > new-en-zh-modal.arpa.zh

如果语言模型创建成功—如何新增语料库词

# 调优****tuning****

# Training the Translation System

nohup nice ~/mouse/md2/scripts/training/train-model.perl -root-dir /home/xiaosun/mouse/compile/woring/en-zh -corpus ~/mouse/compile/corpus/en-zh/TED2013.en-zh.clean -f en -e zh -alignment grow-diag-final-and -reordering msd-bidirectional-fe -lm 0:3:/home/xiaosun/mouse/compile/languagemodel/en-zh/new-en-zh-modal.arpa.zh:8 -external-bin-dir ~/mouse/compile/tools/training-tools -mgiza -mgiza-cpus 6 &> training.out | tee –a training.out&

**nohup nice ~/mouse/md2/scripts/training/train-model.perl -root-dir /home/xiaosun/mouse/compile/woring/en-zh -corpus ~/mouse/compile/corpus/en-zh/TED2013.en-zh.clean -f en -e zh -alignment grow-diag-final-and-reordering msd-bidirectional-fe -lm 0:3:/home/xiaosun/mouse/compile/languagemodel/en-zh/new-en-zh-modal.arpa.zh:8 -external-bin-dir ~/mouse/compile/tools/training-tools/ -mgiza &>training.out**

**nohup nice /home/xiaosun/mouse/md2/scripts/training/train-model.perl -root-dir /home/xiaosun/mouse/compile/working/en-zh -corpus /home/xiaosun/mouse/compile/corpus/en-zh/TED2013.en-zh.clean -f en -e zh -alignment grow-diag-final-and-reordering msd-bidirectional-fe -lm 0:3:/home/xiaosun/mouse/compile/languagemodel/en-zh/new-en-zh-modal.arpa.zh:8 -external-bin-dir /home/xiaosun/mouse/compile/tools/training-tools -mgiza &> training.out &**

**test:**

**~/mouse/md2/bin/moses -f ~/mouse/compile/working/en-zh/model/moses.ini**

参数说明

培训参数（第8.3节）training data (Section 5.2).

* MGIZA

.../train-model.perl -mgiza -mgiza-cpus 8

external-bin-dir ~/mouse/compile/tools/training-tools –mgiza

snt2cooc.pl

snt2coocrmp

* -cores 4
  + 训练语料库和对齐可以分割和短语对每个部分可以同时提取。这可以通过简单地使用参数内核来实现
* sort -S 10G --batch-size 253 --compress-program(压缩程序) gzip –parallel(并行) 5
* -sort-buffer-size 8G -sort-compress gzip -sort-parallel 6

.../train-model.perl -mgiza -mgiza-cpus 8 -cores 10 \

-parallel -sort-buffer-size 10G -sort-batch-size 253 \

-sort-compress gzip -sort-parallel 10

.../train-model.perl -mgiza -mgiza-cpus 2 -snt2cooc snt2cooc.pl \

-parallel -sort-batch-size 253 -sort-compress gzip

**For servers with older OSes, and therefore older sort commands:**

[TRAINING]

script = $moses-script-dir/training/train-model.perl

training-options = "-mgiza -mgiza-cpus 8 -cores 10 -parallel"

parallel = yes

**For laptops with limited memory:**

[TRAINING]

script = $moses-script-dir/training/train-model.perl

training-options = "-mgiza -mgiza-cpus 2 -snt2cooc snt2cooc.pl \

-parallel -sort-batch-size 253 -sort-compress gzip"

parallel = yes

Incremental Training（增量训练）

training-options = "-final-alignment-model hmm"

编译摩西之前你必须: --with-mm

% zcat ${CORPUS}.${L1}.gz | mtt-build -i -o /some/path/${CORPUS}.${L1}

% zcat ${CORPUS}.${L2}.gz | mtt-build -i -o /some/path/${CORPUS}.${L2}

% zcat ${CORPUS}.${L1}-${L2}.symal.gz | symal2mam /some/path/${CORPUS}.${L1}-${L2}.mam

% mmlex-build /some/path/${CORPUS} ${L1} ${L2} -o /some/path/${CORPUS}.${L1}-${L2}.lex

+

oldTrPrbs: <path-to-original-thmm>

oldAlPrbs: <path-to-original-hhmm>

训练步骤

公共参数

-cores cpu核心数

### Prepare data (45 minutes)

参数:

### 2. Run GIZA++ (16 hours)

参数:

--parallel 并行处理

-external-bin-dir ~/mouse/compile/tools/training-tools -mgiza -mgiza-cpus 6

### 3. Align words (2:30 hours)

参数:

--alignment 字词对齐

### 4. Get lexical translation table (30 minutes)

参数:

### 5. Extract phrases (10 minutes)

参数:

### 6. Score phrases (1:15 hours)

参数:

### 7. Build lexicalized reordering model (1 hour)

参数:

• Modeltype - the type of model used (see above)

– wbe - word-based extraction (but phrase-based at decoding). This is the original

model in Moses. DEFAULT

– phrase - phrase-based model

– hier - hierarchical model

• Orientation - Which classes of orientations that are used in the model

– mslr - Considers four different orientations: monotone, swap, discontinuous-left,

discontinuous-right

– msd - Considers three different orientations: monotone, swap, discontinuous (the

two discontinuous classes of the mslr model are merged into one class)

– monotonicity - Considers two different orientations: monotone or non-monotone

(swapanddiscontinuousofthemsdmodelaremergedintothenon-monotoneclass)

– leftright - Considers two different orientations: left or right (the four classes

in the mslr model are merged into two classes, swap and discontinuous-left into

left and monotone and discontinuous-right into right)

• Directionality - Determines if the orientation should be modeled based on the previous

or next phrase, or both.

– backward - determine orientation with respect to previous phrase DEFAULT

– forward - determine orientation with respect to following phrase

– bidirectional - use both backward and forward models

• language - decides which language to base the model on

– fe - conditioned on both the source and target languages

– f - conditioned on the source language only

• collapsing - determines how to treat the scores

– allff - treat the scores as individual feature functions DEFAULT

– collapseff - collapse all scores in one direction into one feature function

any possible configuration of these five factors is allowed. It is always necessary to specify

orientation and language. The other three factors use the default values indicated above if they

are not specified. Some examples of possible models are:

• msd-bidirectional-fe (this model is commonly used, for instance it is the model used

in the WMT baselines 3 )

• wbe-msd-bidirectional-fe-allff same model as above

• mslr-f

• wbe-backward-mslr-f-allff same model as above

• phrase-msd-bidirectional-fe

• hier-mslr-bidirectional-fe

• hier-leftright-forward-f-collapseff

-reordering distance

-reordering msd-bidirectional-fe

-reordering msd-bidirectional-fe,hier-mslr-bidirectional-fe

-reordering distance,msd-bidirectional-fe,hier-mslr-bidirectional-fe

• --reordering-smooth - specifies the smoothing constant to be used for training lexical-

ized reordering models. If the letter u follows the constant, smoothing is based on actual

counts. (default 0.5)

• --max-lexical-reordering - if this flag is used, the extract file will contain information

for the mslr orientations for all three model types, wbe, phrase and hier. Otherwise

the extract file will contain the minimum information that is needed based on which

reordering model config strings that are given.

### 8. Build generation models

参数:

### 9. Create configuration file (1 second)

参数:

# Moses server 配置

**~/mouse/md2/bin/moses --server --daemon -f ~/mouse/compile/working/en-zh/model/moses.ini --server-port 10010**

# EMS系统Experiment Management System

Find experiment.perl in scripts/ems

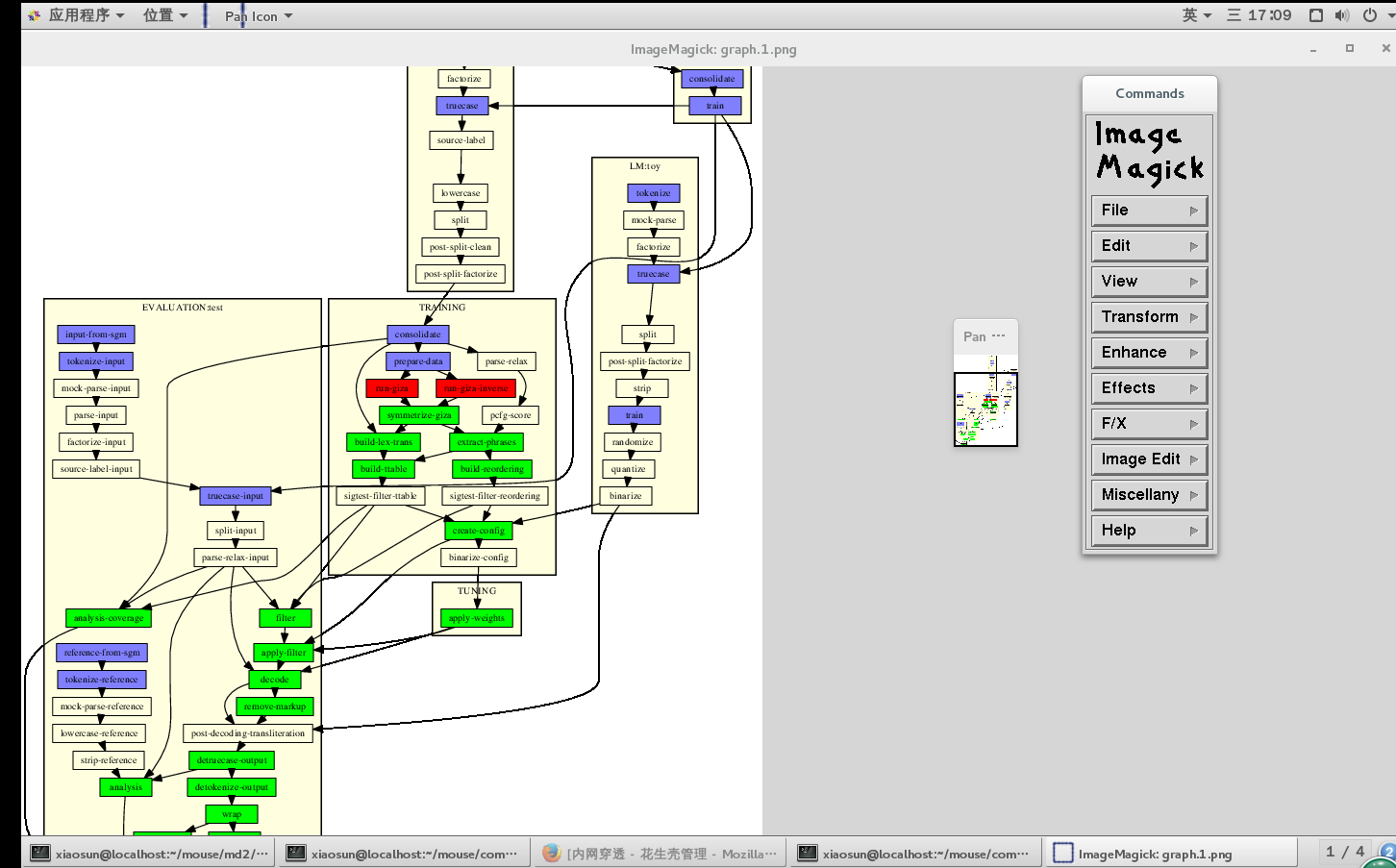
experiment.perl -config config.toy

配置文件中mgiza: training-options = "-mgiza -mgiza-cpus 4"

* working-dir
* data-dir
* moses-script-dir
* moses-src-dir
* srilm-dir
* decoder
* external-bin-dir

experiment.perl -config config.toy -exec.

nohup nice ~/mosesdecoder/scripts/ems/experiment.perl -config config -exec &> log &



# lampp 安装

# Training Syntax Models

# 参数

Training 训练参数 Training Step 6: Score Phrases

* 1. --OnlyDirect：Only creates a model with direct conditional probabilities p(f|e) instead of the default direct and indirect (p(f|e) and p(e|f))
  2. --MaxSpan SIZE: maximum span size of the rule. Default is 15.
  3. --MaxSymbolsSource SIZEand--MaxSymbolsTarget SIZE:Whilearulemaybeextracted

from a large span, much of it may be knocked out by sub-phrases that are substituted by non-terminals. So, fewer actual symbols (non-terminals and words remain). The default maximum number of symbols is 5 for the source side, and practically unlimited (999) for the target side.

* 1. --MinWords SIZE: minimum number of words in a rule. Default is 1, meaning that each

rule has to have at least one word in it. If you want to allow non-lexical rules set this to zero. You will not want to do this for hierarchical models.

* 1. --AllowOnlyUnalignedWords: This is related to the above. A rule may have words in it,

but these may be unaligned words that are not connected. By default, at least one aligned word is required. Using this option, this requirement is dropped.

* 1. --MaxNonTerm SIZE: the number of non-terminals on the right hand side of the rule. This has an effect on the arity of rules, in terms of non-terminals. Default is to generate only binary rules, so the setting is 2.
  2. --MinHoleSource SIZE and --MinHoleTarget SIZE: When sub-phrases are replaced by

non-terminals, we may require a minimum size for these sub-phrases. The default is 2 on the source side and 1 (no limit) on the target side.

* 1. --DisallowNonTermConsecTarget and --NonTermConsecSource. We may want to re-

strict if there can be neighboring non-terminals in rules. In hierarchical models there

is a bad effect on decoding to allow neighboring non-terminals on the source side. The

default is to disallow this -- it is allowed on the target side. These switches override the

defaults.

1.9 --NoFractionalCounting: For any given source span, any number of rules can be gen-

erated. By default, fractional counts are assigned, so probability of these rules adds up to one. This option leads to the count of one for each rule.

2.0 --NoNonTermFirstWord: Disallows that a rule starts with a non-terminal.

Once rules are collected, the file of rules and their counts have to be converted into a proba-bilistic model. This is called rule scoring, and there are also some additional options:

2.1 --OnlyDirect: onlyestimatesdirectconditionalprobabilities. Notethatthisoptionneeds

to be specified for both rule extraction and rule scoring.

2.2 --NoLex: only includes rule-level conditional probabilities, not lexical scores.

2.3 --GoodTuring: Uses Good Turing discounting to reduce actual accounts. This is a good

thing, use it.

Optimizing Moses（优化摩西）

参数加在：[启动server 或者测试语言模型中](#_Moses_server_配置)

1）--threads 6 多线程

2）转二进制文件

Options when using compact phrase and reordering tables.:

--minphr-memory Load phrase table in minphr format into

memory

--minlexr-memory Load lexical reordering table in

minlexr format into memory

Training 训练语言模型参数train-model.perl

--root-dir -- root directory, where output files are stored

存放输出文件的根目录

--corpus -- corpus file name (full pathname), excluding extension

语料库文件名 (全路径名)，不包含扩展

--corpus-dir -- corpus directory (default $ROOT/corpus)

--e -- extension of the English corpus file

英文语料库的拓展文件

--f -- extension of the foreign corpus file

外文语料库的拓展文件

--lm -- language model: <factor>:<order>:<filename> (option can be repeated)

语言模型: <factor>:<order>:<filename>(选项可重复)

--first-step -- first step in the training process (default 1)

--last-step -- last step in the training process (default 7)

--parts -- break up corpus in smaller parts before GIZA++ training

--lexical-dir -- lexical translation probability directory (default $ROOT/model)

--model-dir -- model directory (default $ROOT/model)

--extract-file -- extraction file (default $ROOT/model/extract)

• --corpus-dir -- corpus directory (default $ROOT/corpus)

• --lexical-dir -- lexical translation probability directory (default $ROOT/model)

• --model-dir -- model directory (default $ROOT/model)

• --extract-file -- extraction file (default $ROOT/model/extract)

--giza-f2e -- GIZA++ directory (default $ROOT/giza.$F-$E)

• --giza-e2f -- inverse GIZA++ directory (default $ROOT/giza.$E-$F)

• --alignment -- heuristic used for word alignment: intersect, union, grow, grow-final,

grow-diag, grow-diag-final (default), grow-diag-final-and, srctotgt, tgttosrc

• --max-phrase-length -- maximum length of phrases entered into phrase table (default

7)

• --giza-option -- additional options for GIZA++ training

• --verbose -- prints additional word alignment information

• --no-lexical-weighting -- only use conditional probabilities for the phrase table, not

lexical weighting

• --parts -- prepare data for GIZA++ by running snt2cooc in parts

• --direction -- run training step 2 only in direction 1 or 2 (for parallelization)

• --reordering -- specifies which reordering models to train using a comma-separated

list of config-strings, see FactoredTraining.BuildReorderingModel (Section 5.10). (default

distance)

• --reordering-smooth -- specifies the smoothing constant to be used for training lexi-

calized reordering models. If the letter "u" follows the constant, smoothing is based on

actual counts. (default 0.5)

• --alignment-factors --

• --translation-factors --

• --reordering-factors --

• --generation-factors --

• --decoding-steps --

**-alignment 子参数**

• intersect -- the intersection of the two GIZA++ alignments is taken. This usually creates

a lot of extracted phrases, since the unaligned words create a lot of freedom to align

phrases.

• union -- the union of the two GIZA++ alignments is taken

• grow-diag-final -- the default heuristic

• grow-diag -- same as above, but without a call to function FINAL() (see background to

word alignment).

• grow -- same as above, but with a different definition of neighboring. Now diagonally

adjacent alignment points are excluded.

• grow -- no diagonal neighbors, but with FINAL()

Different heuristic may show better performance for a specific language pair or corpus, so some

experimentation may be useful.

Summary

• --alignment -- heuristic used for word alignment: intersect, union, grow, grow-final,

grow-diag, grow-diag-final (default)

1、所有training参数

1.1 基本选项

--root-dir-- 存放输出文件的根目录

--corpus-- 语料库文件名 (全路径名)，不包含扩展

--e-- 英文语料库的拓展文件

--f-- 外文语料库的拓展文件

--lm-- 语言模型: <factor>:<order>:<filename>(选项可重复)

1.2 翻译模型设置

--alignment-factors--

--translation-factors--

--reordering-factors--

--generation-factors--

--decoding-steps--

1.3 词汇化语法重排模型

--reordering-- 指定重排模型训练一个使用以逗号分隔的config-strings的列表, 参见FactoredTraining.BuildReorderingModel.(default distance)

--reordering-smooth-- 指定参数用在训练词汇化语法重新排序模型。如果字母"u" 跟随着常数, 平滑基于实际数量。(default 0.5)

1.4 部分training

因为MOSES是模块化的可以只使用部分功能，而使用更优异的其他方法

1. 准备数据

2. 运行GIZA++

3. 字对齐

4. 得到词汇转换表

5. 提取短语

6. 短语评分

7. 构建重排序模型

8. 构建生成模型

9. 创建配置文件

--first-step-- 训练过程的第一步 (default 1)

--last-step-- 训练过程的最后一步 (default7)

1.5 文件路径

--corpus-dir-- 语料库的目录 (default$ROOT/corpus)

--lexical-dir-- 词汇翻译概率的目录 (default$ROOT/model)

--model-dir-- 模型目录 (default$ROOT/model)

--extract-file-- 抽取的文件 (default$ROOT/model/extract)

--giza-f2e-- GIZA++ 目录 (default$ROOT/giza.$F-$E)

--giza-e2f-- 倒转的 GIZA++ 目录 (default $ROOT/giza.$E-$F)

1.6 对齐的启发式

--alignment-- 用于词对齐的启发式:intersect, union, grow, grow-final, grow-diag, grow-diag-final (default),grow-diag-final-and, srctotgt, tgttosrc

intersect-- 两个GIZA++算法使用后的交集。这通常产生大量被提取的短语, 因为未对齐短语产生大量自由对齐的短语。

union-- 两个GIZA++算法使用后的并集。

grow-diag-final-- 默认的启发式

grow-diag-- 同上，但是不运行FINAL()功能

grow-- 同上，但是使用不同的临近定义。现在对角相邻对齐点被排除在外。 grow -- 没有对角相邻对齐点，但是有FINAL()

1.7 最大短于长度

--max-phrase-length-- 加入短于表中的最大短于长度(default7)

1.8 GIZA++选项

--giza-option-- GIZA++ 训练的额外选项

1.9 大型训练语料的处理

--parts-- 通过部分运行snt2cooc为GIZA++准备数据

--direction-- 仅在1、2的指导下运行训练步骤 2  (并行化的)

1.10 其他

--verbose-- 打印附加的词对齐信息

--no-lexical-weighting-- 对于短语表仅使用条件概率不使用词汇权重

Factored Training分子训练？

--alignment-factors FACTORMAP

--translation-factors FACTORMAPSET

--reordering-factors FACTORMAPSET

--generation-factors FACTORMAPSET

--decoding-steps LIST

处理大型语料？

• --parts -- prepare data for GIZA++ by running snt2cooc in parts

• --direction -- run training step 2 only in direction 1 or 2 (for parallelization)

# 训练记录

En-zh 联合国语料库254460.432

语料大小: 15886041(en) 15886041(zh)

**14616000消耗52g内存**

**硬盘空间:**

**语料库:**

**语言模型:11G**

**工作空间:**

所用时间:

所用内存:

# 参考

1. 训练，解码 内存不足 4.3
2. add a new feature function to the decoder? Section 7.6).
3. Dealing with large training corpora

###############################moses框架结构############################

a)　总调用部分（主函数）：moses-cmd/src/Main.cpp （The main function）  
　b)　解码器初始入口部分（Initialize the decoder）：  
　　　 i.　参数设置：  
　　　　　moses/src/Parameter.cpp （specifies parameters）  
　　　ii.　模型数据加载：  
　　　　　moses/src/StaticData.cpp （contains globals, loads tables）  
　c)　句子翻译处理部分（Process a sentence）：  
　　　i.　解码器算法实现总调度：  
　　　　　Manager.cpp （implements the decoding algorithm）  
　　　ii.　翻译选项处理：  
　　　　　TranslationOptionCollection.cpp （contains translation options）  
　　　iii.　部分翻译处理：  
　　　　　Hypothesis.cpp（represents partial translation）？  
　　　iv.　包含翻译假设，实现剪枝：  
　　　　　HypothesisStack.cpp （contain viable hypotheses, implements pruning）  
　d)　结果输出：moses-cmd/src/Main.cpp（Output results）  
　　　i.　输出最佳翻译结果：  
　　　　　moses-cmd/src/IOStream::OutputBestHypo （print best translation）  
　　　ii.　n-best生成及输出：  
　　　　（n-best lists generated in Manager.cpp, output in IOStream.cpp）###############################moses框架结构############################

# 问题

分割训练

Training Step 6: Score Phrases

第七步使用

能否在磁盘中训练

**grow-diag-final-and-reordering 训练过程中的参数？**

**语料合并-🡪重头开始**

处理大型语料？

磁盘

磁盘空间扩展