NiuTrans 开源统计机器翻译系统
CWMT2013 基线系统构建文档

东北大学自然语言处理实验室
NiuTrans 团队
niutrans@mail.neu.edu.cn
http://www.nlplab.com
http://weibo.com/niutrans
http://www.niutrans-bbs.com
http://183.129.153.66:81/forum.php (niutrans-bbs by ip)

目录

为了有效的帮助参评单位搭建自己的翻译系统，NiuTrans 团队提供了基线系统的详细搭建步骤。在使用 NiuTrans 构建翻译系统之前，建议您阅读 NiuTrans_CWMT 系统使用说明 http://www.nlplab.com/NiuPlan/NiuTrans.ch.html。如有问题，请到小牛论坛 www.niutrans-bbs.com 或 http://183.129.153.66:81/forum.php 中 CWMT2013 专区进行讨论，或发送邮件至 niutrans@mail.neu.edu.cn。

1 汉英短语基线翻译系统

整个汉英短语翻译基线系统搭建共分为十二步，这些步骤包括：数据使用，数据预处理，词对齐，训练翻译模型，训练语言模型，生成配置文件，规则过滤，权重调优，解码，恢复大小写，detokenizer，评价。在这些步骤中训练翻译模型和训练语言模型可以并行操作，其他步骤需要按顺序来执行。

1.1 数据使用

评测组织方发放的新闻数据有三种，分别为：训练集、开发集、往年评测数据。其中，训练集是纯文本格式，编码：UTF-8(无 BOM),开发集和往年评测数据是 xml 格式(UTF8)，需要把开发集和往年评测数据从 xml 里面抽出来，处理成一行中文一行英文的纯文本格式 UTF8(无 BOM)。

汉英短语基线系统构建的时候，把往年的评测数据一部分作为测试数据用来评价 NiuTrans.Phrase 的性能，一部分用作训练数据。

训练集是官方提供的训练数据加上 2003 年 863 机器翻译评测数据。

开发集是官方开发集。

测试集是 2004 年 863 机器翻译评测数据，2005 年机器翻译 863 评测数据，2007 年 SSMT 机器翻译评测数据。

1.1.1 训练集

训练集包含以下数据，每个文件夹里只有双语句对，文件格式:UTF8(无 BOM)。存放格式如下：

```
data/train                          #训练集
  |-- Datum                         #点通数据集
  |    |-- c.input.txt              #中文句对
  |    |-- e.input.txt              #英文句对
  |-- HIT-IR
  |-- HIT-MT
  |-- ICT-web-A
  |-- ICT-web-B
  |-- NEU
```

```
|-- XMU-Movie-Subtitle
|-- CLDC-LAC-2003-006
|-- CLDC-LAC-2003-004
|--2003-863-001
```

1.1.2 开发集

默认开发集：汉英 1006 句，英汉 1000 句。从原始 xml 抽取出来，保存为 1 个源语，4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下:

```
data/dev                    #开发集
  |--C2E                    #汉英开发集
  |    |-- c.input.txt      #中文句对
  |    |-- e.input.ref1.txt #中文句对翻译 1
  |    |-- e.input.ref2.txt #中文句对翻译 2
  |    |-- e.input.ref3.txt #中文句对翻译 3
  |    |-- e.input.ref4.txt #中文句对翻译 4
  |--E2C
      |-- e.input.txt
      |-- c.input.ref1.txt
      |-- c.input.ref2.txt
      |-- c.input.ref3.txt
      |-- c.input.ref4.txt
```

1.1.3 测试集

选择的测试集有：2004-863-001,2005-863-001,SSMT-2007。从原始 xml 文件抽取出来，保存为 1 个源语 4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下：

```
data/test                          #测试集
  |-- SSMT-2007                    #SSMT-2007 测试集
  |      |-- C2E                   #SSMT-2007 汉英测试集
  |      |      |-- c.input.txt    #中文句对
  |      |      |-- e.input.ref1.txt #中文句对翻译 1
  |      |      |-- e.input.ref2.txt #中文句对翻译 2
  |      |      |-- e.input.ref3.txt #中文句对翻译 3
  |      |      |-- e.input.ref4.txt #中文句对翻译 4
  |      |-- E2C
  |             |-- c.input.txt
  |             |-- e.input.ref1.txt
  |             |-- e.input.ref2.txt
  |             |-- e.input.ref3.txt
```

```
|              |-- e.input.ref4.txt
|-- 2005-863-001                #同 SSMT-2007
|-- 2004-863-001                #同 SSMT-2007
```

1.1.4 语言模型训练文件

非评测组织方提供的可用训练数据：路透社语料库和搜狗全网新闻语料库。数据存放格式如下：

```
data/LM                         #语言模型
  |-- sougou.txt                #搜狗中文单语语料
  |-- router.txt                #路透社英文单语语料
```

汉英短语系统最终用到的语言模型训练文件是路透社语料库加上训练集所有英文语料。

1.2 数据预处理

数据预处理是统计机器翻译系统的第一步，训练集，开发集，测试集，语言模型文件都需要通过数据预处理。NiuTrans.Phrase 预处理主要包括：乱码过滤，分词，泛化一些数词、时间词、日期词，翻译数词、时间词、日期词等，不同数据处理的步骤也不一样，下面给出本次基线系统数据预处理用到的脚本和不同数据集使用的处理步骤。

为了方便的给出示例，假设所有数据在 NiuTrans_CWMT 目录的 data 文件夹里，里面有文件夹：训练集(train)，存储内容和格式见 1.1 训练集；开发集(dev)，存储内容和格式见 1.2 开发集；测试集(test)，存储内容和格式见 1.3 测试集；语言模型(LM)，包含路透社语料和搜狗语料。

1.2.1 双语乱码过滤

脚本：NiuTrans-clear.illagal.char.pl。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-clear.illegal.char.pl \
         -src     中文文本(无论训练汉英/英汉翻译系统，该参数都指定为中文) \
         -tgt     英文文本(无论训练汉英/英汉翻译系统，该参数都指定为英文) \
         -outSrc  输出乱码过滤后中文文本 \
         -outTgt  输出乱码过滤后英文文本
```

示例：部分训练数据进行双语乱码过滤的命令

```
$> perl   NiuTrans-clear.illegal.char.pl \
          -src      ../data/train/NEU/c.input.txt \
          -tgt      ../data/train/NEU/e.input.txt \
          -outSrc   ../data/train/NEU/c.input.txt.clean \
          -outTgt   ../data/train/NEU/e.input.txt.clean
```

1.2.2 单语乱码过滤

脚本：NiuTrans-monolingual.clear.illegal.char.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      待处理文本 \
          -outTgt   乱码过滤后输出文本 \
          -lang     语言类别(中文:zh  英文:en)
示例：语言模型(搜狗单语)进行单语乱码过滤的命令
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      ../data/LM/sogou.txt \
          -outTgt   ../data/LM/sogou.txt.clean \
          -lang     zh
```

注：双语过滤和单语过滤，过滤掉的乱码存放在 .discard 文件中。

1.2.3 预处理分词

```
$> perl   NiuTrans-running-segmenter.pl \
          -lang     ch \
          -input    输入待处理文本，这里指乱码过滤后文本 \
          -output   输出预处理分词后文本 \
          -method   泛化翻译开关(00/01/11)
示例：本次基线系统部分训练数据的分词命令及参数
$> perl   NiuTrans-running-segmenter.pl \
          -lang     ch \
          -input    ../data/train/NEU/c.input.txt.clean \
          -output   ../data/train/NEU/c.input.txt.clean.token \
          -method   01
```

脚本：NiuTrans-running-segmenter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入上图

命令。

注：“-lang”为预处理分词文件类型，“ch”表示为中文，“en”表示为英文。

“-method”为预处理分词泛化和翻译功能的开关，00：不泛化也不翻译，01：只泛化不翻译，11：泛化并且翻译。

1.2.4 开发集，测试集格式处理

NiuTrans 系统进行权重调优的时候需要将开发集处理成一个源语，一个空行，四个目标语的格式作为输入，所以开发集经过预处理分词后，还要进行格式处理。

NiuTrans 系统对测试集结果进行官方评价时，需要生成官方工具需要的 xml 文件，生成工具同样需要将测试集处理成一个源语，一个空行，N 个目标语的格式。

脚本：NiuTrans-dev-merge.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-dev-merge.pl \
          源文 \
          译文 1 \
          译文 2 \
          •••
          译文 n   > 标准格式文件
```

示例：本次基线系统开发集格式处理命令及参数

```
$> perl   NiuTrans-dev-merge.pl \
          ../data/dev/c.input.txt.token \
          ../data/dev/e.input.ref1.txt.token \
          ../data/dev/e.input.ref2.txt.token \
          ../data/dev/e.input.ref3.txt.token \
          ../data/dev/e.input.ref4.txt.token      >   dev.txt
```

1.2.5 训练集，开发集，测试集，语言模型处理的步骤

本次 Niutrans 基线系统中数据预处理的处理步骤如下:

1 训练集：首先进行双语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。

2 开发集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化

开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行开发集格式处理，处理成一个源语，一个空行和四个目标语的格式。

3 测试集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行测试集格式处理，处理成一个源语，一个空行和四个目标语的格式。

4 语言模型源文件：首先进行单语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。最终使用的语言模型训练文件是处理好的训练集目标语文件与相应的单语语言模型训练文件的合并。以汉英翻译系统为例，语言模型训练文件：处理过的训练集英文部分与处理过的路透社文件的合并，也可以先合并在处理。

注：训练集过滤掉行数请参见附件 1。

1.3 词对齐

```
$> perl  NiuTrans-length.ratio.filter.pl \
         -src                源文本分词结果 \
         -tgt                目标文本分词结果 \
         -outSrc             源文本分词后长度比过滤结果 \
         -outTgt             目标文本分词后长度比过滤结果 \
         -lowerBoundCoef 源语长度和目标语长度比值下限 \
         -upperBoundCoef 源语长度和目标语长度比值上限
示例：部分训练集双语乱码过滤后的文件进行长度比过滤的命令
$> perl  NiuTrans-length.ratio.filter.pl \
         -src                ../data/train/NEU/c.input.txt.clean.token \
         -tgt                ../data/train/NEU/e.input.txt.clean.token \
         -outSrc             ../data/train/NEU/c.input.txt.clean.token.filter \
         -outTgt             ../data/train/NEU/e.input.txt.clean.token.filter \
         -lowerBoundCoef 0.4 \
         -upperBoundCoef 1.8
```

本次 NiuTrans 基线系统的训练集词语对齐采用的是开源工具：GIZA++1.0.7。

下载链接：http://code.google.com/p/giza-pp/downloads/detail?name=giza-pp-v1.0.7.tar.gz。

NiuTrans 对该工具进行了封装(用 perl 调用 GIZA++工具)。

训练集词对齐分为两步：首先进行长度比过滤，然后再进行词对齐训练。

训练语料词语对齐的步骤：

1. 对训练语料源语言/目标语言分词结果进行长度比过滤（原文和译文的长度比例限制和最大长度限制）。

长度比过滤脚本：NiuTrans-length.ratio.filter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入上图中命令

注：长度比过滤掉的文件是.discard，过滤掉的行数请参见附件 1。

2. 采用 Niutrans 词对齐脚本跑词对齐（仅支持 Linux 平台）。

词对齐脚本：NiuTrans-running-GIZA++.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                    -src      源文本长度比结果 \
                    -tgt      目标文本长度比结果 \
                    -out      词对齐文件 \
                    -tmpdir   生成临时文件的目录
示例：部分训练集长度比过滤的文件跑词对齐的命令
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                    -src      ../data/train/NEU/c.input.txt.clean.token \
                    -tgt      ../data/train/NEU/e.input.txt.clean.token \
                    -out      ../data/train/NEU/alignment.txt \
                    -tmpdir   ../work/tmp/ &
```

注：由于长度比过滤后的训练集文件较大(总共 450w 左右),跑词对齐时可以分割成 9 个文件，每个 50w 左右，以减少词对齐训练时间。

最后在 train 文件夹里面创建一个 all 文件夹，再把 train 下每个文件夹下的源语、目标语的长度比过滤结果和对应的词对齐文件纵向合并到一起(源语之间合并、目标语之间合并、词对齐之间合并)，分别放到 all 里面(c.input.txt.clean.token ,e.input.txt.clean.token,alignment.txt)，作为训练翻译模型的输入文件。

1.4 训练翻译模型

所有训练数据经过数据预处理、词对齐训练，合并后，基线系统就可以训练翻译模型了。为了使训练翻译模型变得简单，NiuTrans.Phrase 系统默认带有一个训练翻译模型的配置文件。训练模型时可以先修改一下配置文件，然后训练翻译模型，也可以按默认配置训练翻译模型。

训练翻译模型的配置文件：NiuTrans.phrase.train.model.config，在NiuTrans_CWMT\config 目录下。

1．本次 Niutrans.Phrase 汉英翻译基线系统的配置文件修改了如下参数，如下图：

```
##########################################
### NiuTrans  phrase train model config ###
##########################################

# temp file path
param="Lexical-Table"                value="lex"
param="Extract-Phrase-Pairs"         value="extractPhrasePairs"

# phrase table parameters
param="Max-Source-Phrase-Size"       value="7"
param="Max-Target-Phrase-Size"       value="7"
param="Phrase-Cut-Off"               value="0"

# phrase translation model
param="Phrase-Table"                 value="phrase.translation.table"

# maxent lexicalized reordering parameters
param="ME-max-src-phrase-len"        value="7"
param="ME-max-tar-phrase-len"        value="7"
param="ME-null-algn-word-num"        value="1"
param="ME-max-sample-num"            value="5000000"
param="ME-use-src-parse-pruning"     value="0"
param="ME-src-parse-path"            value="/path/to/parse.tree"

# maxent lexicalized reordering model
param="ME-Reordering-Table"          value="me.reordering.table"

# msd lexicalized reordering parameters
param="MSD-model-type"               value="1"

# msd lexicalized reordering model
param="MSD-Reordering-Model"         value="msd.reordering.table"
```


修改了如下参数：

Max-Source-Phrase-Szie 值改为 3

Max-Target-Phrase-Size 值改为 5

ME-max-src-phrase-len 值改为 3

ME-max-tar-phrase-len 值改为 5

ME-max-sample-num 值改为 10000000

2．训练翻译模型的配置文件参数说明：

参数	含义
Lexical-Table	词汇翻译表（临时文件）
Extract-Phrase-Pairs	短语对集合文件（临时文件）

Max-Source-Phrase-Size	短语对源语言端短语长度
Max-Target-Phrase-Size	短语对目标语言端短语长度
Phrase-Table	短语翻译表
ME-max-src-phrase-len	ME 调序模型源语言端短语长度
ME-max-tar-phrase-len	ME 调序模型目标语言端短语长度
ME-max-sample-num	ME 模型训练抽取 sample 个数
ME-Reordering-Table	ME 调序模型
MSD-model-type	MSD 调序模型类型
MSD-Reordering-Model	MSD 调序模型

3．训练翻译模型的脚本：NiuTrans-phrase-train-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-phrase-train-model.pl \
          -tmdir    生成翻译规则表的目录 \
          -s        源语言分词文件 \
          -t        目标语言分词文件 \
          -a        源语言到目标语言的词对齐文件
示例: 本次汉英短语基线系统训练翻译模型的命令
$> mkdir ../work/model   -p
$> perl   NiuTrans-phrase-train-model.pl \
          -tmdir    ../work/model/ \
          -s        ../data/train/all/c.input.txt.clean.token\
          -t        ../data/train/all/e.input.txt.clean.token\
          -a        ../data/train/all/alignment.txt
```

想了解更多参数详参见 http://www.nlplab.com/NiuPlan/NiuTrans.Phrase.ch.html

2.训练翻译模型和高级设置。

注：训练 ME 调序模型需要较大的内存，如果程序运行过程中出现异常，建议减小 ME-max-sample-num 参数值。

1.5 训练语言模型

本次基线系统的语言模型训练使用的是 NiuTrans 语言模型训练工具。NiuTrans.Phrase 汉英基线系统的语言模型源文件是预处理过的路透社英文单语语料和预处理过的所有训练集英文语料合并到一起。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$>cat    ../data/train/all/e.input.txt.clean.token ../data/LM/router.txt.clean.token >
```

```
../data/LM/e.lm.merged.txt
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   语言模型文件 \
          -ngram    n 指 N 元语言模型 \
          -vocab    生成的目标语端语言模型词汇表 \
          -lmbin    生成的目标语端语言模型的二进制文件
示例：本次汉英短语基线系统训练语言模型的命令
mkdir ../work/lm/ -p
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   ../data/LM/e.lm.merged.txt \
          -ngram    5 \
          -vocab    ../work/lm/lm.vocab \
          -lmbin    ../work/lm/lm.trie.data
```

1.6 生成配置文件

NiuTrans.Phrase 基线系统将解码需要用的参数和翻译模型，调序模型写到配置文件里面，供解码时调用。

生成配置文件的脚本：Niutrans-phrase-generate-mert-config.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl    NiuTrans-phrase-generate-mert-config.pl \
          -tmdir    翻译模型文件夹的位置 \
          -lmdir    语言模型文件夹的位置 \
          -ngram    n(代表 N 元语言模型) \
          -o        生成的配置文件（用于解码）
示例：本次汉英短语基线系统生成解码器配置文件的命令
$> mkdir   ../work/config -p
$> perl    NiuTrans-phrase-generate-mert-config.pl \
          -tmdir    ../work/model/ \
          -lmdir    ../work/lm/ \
          -ngram    5 \
          -o        ../work/config/NiuTrans.phrase.user.config
```

本次基线系统解码器配置文件参数如下图：

```
#>>> runtime parameters

# number of MERT iterations
param="nround"                          value="12"
# order of n-gram language model
param="ngram"                           value="5"
# maximum phrase length
param="maxphraselength"                 value="7"
# use punctuation pruning (1) or not (0)
param="usepuncpruning"                  value="1"
# use cube-pruning (1) or not (0)
param="usecubepruning"                  value="1"
# use maxent reordering model (1) or not (0)
param="use-me-reorder"                  value="1"
# use msd reordering model (1) or not (0)
param="use-msd-reorder"                 value="1"
# number of threads
param="nthread"                         value="8"
# how many translations are dumped
param="nbest"                           value="30"
# output OOV words
param="outputoov"                       value="1"
# output source words that are explicitly deleted
param="outputnull"                      value="0"
# beam size (or beam width)
param="beamsize"                        value="30"
# beam scale. This parameter controls the number of hypotheses
# that are computed in decoding.
# e.g., beamscale=3 means we search over 3 * beamsize hyphotheses
param="beamscale"                       value="1"
# number of references of dev. set
param="nref"                            value="4"
# allow null-translation (i.e. word-deletion)
param="usenulltrans"                    value="0"
# allow sequence of null-translations
param="snulltrans"                      value="1"
# normalize output (1) or not (0)
param="normalizeoutput"                 value="0"
# distortion limit
param="maxdd"                           value="10"
# lowercase translation result
param="lowertext"                       value="1"
```

解码器配置文件参数说明：

参数	含义
Ngram-LanguageModel-File	语言模型二进制文件
Target-Vocab-File	语言模型词汇表
ME-Reordering-Table	ME 调序模型文件
MSD-Reordering-Model	MSD 调序模型文件
Phrase-Table	短语翻译表文件
Punct-Vocab-File	解码器标点切分文件（可自定义）
nround	MERT 轮数

ngram	语言模型阶数（最好设置为 5）
usepuncpruning	解码标点切分开关（加速解码方法）
usecubepruning	cube-pruning 开关（加速解码方法）
use-me-reorder	使用 ME 模型开关
use-msd-reorder	使用 MSD 模型开关
nthread	线程数
nbest	生成翻译候选个数
outputoov	输出 OOV 开关
outputnull	输出空翻译开关
beamsize	解码 beam 大小
nref	开发集参考答案个数
usenulltrans	使用空翻译开关
maxdd	最大调序距离
lowertext	解码过程英文小写化
weights	权重（每一维特征使用空格切分）
ranges	开发集特征值调整范围

1.7 规则过滤

由于训练生成翻译模型和调序模型比较大，无法全部转载到内存中，因为调优和解码是在开发集和测试集上，所以用开发集和测试集过滤翻译模型和调序模型。本次 NiuTrans.Phrase 汉英基线系统将开发集和测试集合并到一起过滤翻译模型和 MSD 调序模型。

规则过滤用到的程序：NiuTrans.PhraseExtractor，脚本:filter.msd.model.pl

使用说明：首先进入 NiuTrans_CWMT/目录，然后在命令行下输入：

```
$> bin/NiuTrans.PhraseExtractor --FILPD \
        -dev      开发集和测试集 \
        -in       短语翻译表 \
        -out      过滤后的短语翻译表 \
        -maxlen   翻译表中的最大长度 \
        -rnum     开发集中答案的个数
$> perl   scripts/filter.msd.model.pl \
        过滤后的短语翻译表  \
        MSD 调序表 > 过滤后的 MSD 调序表
示例：本次汉英短语基线系统过滤翻译模型和 MSD 调序模型的命令
$> cat   data/dev/dev.txt   data/test/test.txt > data/dev/dev_and_test.txt
$> bin/NiuTrans.PhraseExtractor --FILPD \
        -dev      data/dev/dev_and_test.txt \
```

```
        -in       work/model/phrase.translation.table \
        -out      work/model/phrase.translation.table.filterDevAndTest \
        -maxlen   10 \
        -rnum     4

$> perl   scripts/filter.msd.model.pl \
        work/model/phrase.translation.table.filterDevAndTest \
        work/model/msd.reordering.table > work/model/msd.reordering.table.filterDevAndTest
```

注意：规则过滤后需要将翻译表和 MSD 调序表的参数替换为过滤后的。

```
$> vim   ../work/config/NiuTrans.phrase.user.config
param = “MSD-Reordering-Model” value = “../work/model.phrase/msd.reordering.table”
```

改为

```
param = “MSD-Reordering-Model” value =
“../work/model.phrase/msd.reordering.table.filterDevAndTest”

param = “Phrase-Table”              value = “../work/model.phrase/phrase.translation.table”
```

改为

```
param = “Phrase-Table”              value =
 “../work/model.phrase/phrase.translation.table.filterDevAndTest”
```

注：dev.txt 是处理好的标准格式(一个源语 N 个目标语)的开发集,test.txt 是处理成标准格式的测试集，这里测试集可以指定 04 年，05 年的 863 评测数据，07 年 SSMT 数据。

详细说明参见：http://www.nlplab.com/NiuPlan/NiuTrans.Phrase.ch.html 5.规则过滤。

1.8 权重调优

NiuTrans.Phrase baseline 直接在处理好的标准开发集上调优，调优后把最佳权重写到配置文件里面，解码时，直接用开发集里 mert 后的最优权重。

权重调优脚本：NiuTrans-phrase-mert-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-phrase-mert-model.pl \
        -dev      处理好的开发集(1 个汉语一个空行 N 个目标语的格式) \
        -c        配置文件 \
        -nref     n(参考答案的个数) \
        -r        mert 轮数（1 轮=12 个 mert 循环） \
        -l        生成调优文件的日志
```

示例：本次汉英短语基线系统进行调优的命令

```
$> perl   NiuTrans-phrase-mert-model.pl \
```

```
        -dev    ../data/dev/dev.txt \
        -c      ../work/config/NiuTrans.phrase.user.config \
        -nref   4 \
        -l      ../work/mert-model.log
```

输出：最优的特征权重被记录在配置文件“NiuTrans_CWMT/work/config/NiuTrans.phrase.user.config”中。这些特征权重将被用于对测试句的解码过程中。

调优后的 weights：value="0.9369 1.3736 0.3712 0.3873 0.2479 0.1654 0.5579 0.78 0 0.8048 0 0.305 0.1851 0.2603 0.3252 0.1468 0.3"。

注：想设置更多权重调优命令，请参见脚本 usage 用法：perl Niutrans-phrase-mert-model.pl，然后回车。

1.9 解码

解码这一步在调优之后，直接利用开发集调优好的权重，在测试集上解码。

解码脚本：perl Niutrans-phrase-decoder-model.pl 输入：测试集，配置文件，输出：1best.out

使用说明：首先进入 NiuTrans_CWMT/script 目录，然后在命令行下输入：

```
$> perl  NiuTrans-phrase-decoder-model.pl \
         -test      测试集 \
         -c         配置文件 \
         -opnull    是否输出 oov(1 是；0 否) \
         -output    翻译结果
示例：本次汉英短语基线系统的解码命令以及使用的参数
$> perl  NiuTrans-phrase-decoder-model.pl \
         -test      ../data/test/test.txt \
         -c         ../work/config/NiuTrans.phrase.user.config \
         -opnull    0 \
         -output    1best.out
```

1.10 恢复大写信息(recasing)

由于测试集是经过预处理分词过的，并且在分词的时候已经把所有大写字母全部转为小写了，所以要对翻译结果恢复大写信息。恢复大写信息需要两步：训练大写信息恢复模型，恢复大写信息。

使用到的脚本：NiuTrans-training-recase-model.pl Niutrans-recaser.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-training-recasing-model.pl \
          -corpus      只做 token 的英文语言模型训练文件 \
          -modelDir    生成 recasing 模型路径

$> perl   NiuTrans-recaser.pl \
          -config      恢复大写信息模型的配置文件 \
          -test        翻译结果 \
          -output      恢复大写信息的翻译结果
示例：本次汉英短语基线系统结果恢复大写信息的命令
$> mkdir  ../work/model.recasing -p
$> perl   NiuTrans-training-recasing-model.pl \
          -corpus      ../data/LM/e.lm.merged.txt \
          -modelDir    ../work/model.recasing/

$> perl   NiuTrans-recaser.pl \
          -config      ../work/config/NiuTrans.phrase.user.config \
          -test        1best.out   \
          -output      1best.out.recased
```

1.11 Detokenizer

这一步对恢复大写的翻译结果进行 detoken 处理。

detoken 脚本：NiuTrans-detokenizer.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   Niutrans-detokenizer.pl \
          -in      恢复大小写的翻译结果 \
          -out     detoken 结果
示例：本次汉英短语基线系统 detoken 命令
$> perl   Niutrans-detokenizer.pl \
          -in      1best.out.recased \
          -out     1best.out.recased.detoken
```

1.12 评价

采用 CWMT2013 官方评价程序，只需将经过后处理的结果变成评价需要的 xml 格式文件，然后输入到官方评价程序即可。这里可以用我们提供 generate-xml-tools（NiuTrans_CWMT/tools/generate-xml-tools）工具生成 xml 文件。

generate-xml.pl 是用来生成 xml 的脚本，使用方法如下：

```
$> perl   generate-xml.pl \
          -flist      系统输出的翻译结果文件 \
          -task       汉英新闻翻译默认为“zh_en_news_trans” \
          -1best      \
          -evaluate \
          -rlist       参考答案文件（标准格式）\
          -nref        参考译文数
示例：
$> perl   generate-xml.pl \
          -flist      1best.out.recased.detoken \
          -task       zh_en_news_trans \
          -1best      \
          -evaluate \
          -rlist      dev.C2E.txt \
          -nref        4
```

运行之后生成三个 xml 文件：

- src.xml：源语言 xml 文件
- ref.xml：参考答案 xml 文件
- 1best.xml： 系统输出翻译结果 xml 文件

最后得到的 xml 文件就是评价需要的文件格式。mteval_sbp 是 CWMT 官方提供的评价工具，我们只要把以上三个文件输入到程序中，就可以得到评价结果 1best_result，命令如下：

mteval_sbp -c -r ref.xml -s src.xml -t 1best.xml >1best_result

本次汉英短语基线系统在开发集上调优后，直接带开发集上解码，恢复大小写，detoken，然后进行官方评价的 BLEU_SBP4 值：

0.2719(不带 **–c** 参数)

0.2577(带 **–c** 参数)

注：汉英短语基线翻译系统的标准配置文件见 NiuTrans_1.3.0_CWMT2013\config\CWMT2013-c2e 目录下：

训练翻译模型的配置文件：NiuTrans.phrase.train.model.CWMT.config

解码器配置文件：NiuTrans.phrase.CWMT.config

2 汉英层次短语基线翻译系统

整个汉英层次短语翻译基线系统搭建共分为十二步，这些步骤包括：数据使用，数据预处理，词对齐，训练翻译模型，训练语言模型，生成配置文件，规则过滤，权重调优，解码，恢复大小写，detokenizer，评价。在这些步骤中训练翻译模型和训练语言模型可以并行操作，其他步骤需要按顺序来执行。

2.1 数据使用

评测组织方发放的新闻数据有三种，分别为：训练集、开发集、往年评测数据。其中，训练集是纯文本格式，编码：UTF-8(无 BOM),开发集和往年评测数据是 xml 格式(UTF8)，需要把开发集和往年评测数据从 xml 里面抽出来，处理成一行中文一行英文的纯文本格式 UTF8(无 BOM)。

汉英层次短语基线系统构建的时候，把往年的评测数据一部分作为测试数据用来评价 NiuTrans.Hierarchy 的性能，一部分用作训练数据。

训练集是官方提供的训练数据加上 2003 年 863 机器翻译评测数据。

开发集是官方开发集。

测试集是 2004 年 863 机器翻译评测数据，2005 年机器翻译 863 评测数据，2007 年 SSMT 机器翻译评测数据。

2.1.1 训练集

训练集包含以下数据，每个文件夹里只有双语句对，文件格式:UTF8(无 BOM)。存放格式如下：

```
data/train                          #训练集
  |-- Datum                         #点通数据集
  |    |-- c.input.txt              #中文句对
  |    |-- e.input.txt              #英文句对
  |-- HIT-IR
  |-- HIT-MT
  |-- ICT-web-A
  |-- ICT-web-B
  |-- NEU
```

```
|-- XMU-Movie-Subtitle
|-- CLDC-LAC-2003-006
|-- CLDC-LAC-2003-004
|--2003-863-001
```

2.1.2 开发集

默认开发集：汉英 1006 句，英汉 1000 句。从原始 xml 抽取出来，保存为 1 个源语，4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下:

```
data/dev                    #开发集
  |--C2E                    #汉英开发集
  |    |-- c.input.txt      #中文句对
  |    |-- e.input.ref1.txt #中文句对翻译 1
  |    |-- e.input.ref2.txt #中文句对翻译 2
  |    |-- e.input.ref3.txt #中文句对翻译 3
  |    |-- e.input.ref4.txt #中文句对翻译 4
  |--E2C
      |-- e.input.txt
      |-- c.input.ref1.txt
      |-- c.input.ref2.txt
      |-- c.input.ref3.txt
      |-- c.input.ref4.txt
```

2.1.3 测试集

选择的测试集有：2004-863-001,2005-863-001,SSMT-2007。从原始 xml 文件抽取出来，保存为 1 个源语 4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下：

```
data/test                          #测试集
  |-- SSMT-2007                    #SSMT-2007 测试集
  |      |-- C2E                   #SSMT-2007 汉英测试集
  |      |      |-- c.input.txt    #中文句对
  |      |      |-- e.input.ref1.txt #中文句对翻译 1
  |      |      |-- e.input.ref2.txt #中文句对翻译 2
  |      |      |-- e.input.ref3.txt #中文句对翻译 3
  |      |      |-- e.input.ref4.txt #中文句对翻译 4
  |      |-- E2C
  |             |-- c.input.txt
  |             |-- e.input.ref1.txt
  |             |-- e.input.ref2.txt
  |             |-- e.input.ref3.txt
```

```
|              |-- e.input.ref4.txt
|-- 2005-863-001                    #同 SSMT-2007
|-- 2004-863-001                    #同 SSMT-2007
```

2.1.4 语言模型训练文件

非评测组织方提供的可用训练数据：路透社语料库和搜狗全网新闻语料库。数据存放格式如下：

```
data/LM                             #语言模型
  |-- sougou.txt                    #搜狗中文单语语料
  |-- router.txt                    #路透社英文单语语料
```

汉英短语系统最终用到的语言模型训练文件是路透社语料库加上训练集所有英文语料。

2.2 数据预处理

数据预处理是统计机器翻译系统的第一步，训练集，开发集，测试集，语言模型文件都需要通过数据预处理。NiuTrans.Phrase 预处理主要包括：乱码过滤，分词，泛化一些数词、时间词、日期词，翻译数词、时间词、日期词等，不同数据处理的步骤也不一样，下面给出本次基线系统数据预处理用到的脚本和不同数据集使用的处理步骤。

为了方便的给出示例，假设所有数据在 NiuTrans_CWMT 目录的 data 文件夹里，里面有文件夹：训练集(train)，存储内容和格式见 1.1 训练集；开发集(dev)，存储内容和格式见 1.2 开发集；测试集(test)，存储内容和格式见 1.3 测试集；语言模型(LM)，包含路透社语料和搜狗语料。

2.2.1 双语乱码过滤

脚本：NiuTrans-clear.illagal.char.pl。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-clear.illegal.char.pl \
         -src      中文文本(无论训练汉英/英汉翻译系统，该参数都指定为中文) \
         -tgt      英文文本(无论训练汉英/英汉翻译系统，该参数都指定为英文) \
         -outSrc   输出乱码过滤后中文文本 \
         -outTgt   输出乱码过滤后英文文本
```

示例：部分训练数据进行双语乱码过滤的命令

```
$> perl   NiuTrans-clear.illegal.char.pl \
          -src      ../data/train/NEU/c.input.txt \
          -tgt      ../data/train/NEU/e.input.txt \
          -outSrc   ../data/train/NEU/c.input.txt.clean \
          -outTgt   ../data/train/NEU/e.input.txt.clean
```

2.2.2 单语乱码过滤

脚本：NiuTrans-monolingual.clear.illegal.char.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      待处理文本 \
          -outTgt   乱码过滤后输出文本 \
          -lang     语言类别(中文:zh 英文:en)
示例：语言模型(搜狗单语)进行单语乱码过滤的命令
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      ../data/LM/sogou.txt \
          -outTgt   ../data/LM/sogou.txt.clean \
          -lang     zh
```

注：双语过滤和单语过滤，过滤掉的乱码存放在 .discard 文件中。

2.2.3 预处理分词

脚本：NiuTrans-running-segmenter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-running-segmenter.pl
          -lang     ch \
          -input    输入待处理文本，这里指乱码过滤后文本 \
          -output   输出预处理分词后文本
          -method   泛化翻译开关(00/01/11)
示例：本次基线系统部分训练数据的分词命令及参数
$> perl   NiuTrans-running-segmenter.pl
          -lang     ch \
          -input    ../data/train/NEU/c.input.txt.clean \
          -output   ../data/train/NEU/c.input.txt.clean.token
          -method   01
```

注：“-lang”为预处理分词文件类型，“ch”表示为中文，“en”表示为英文。

“-method”为预处理分词泛化和翻译功能的开关，00：不泛化也不翻译，01：只泛化不翻译，11：泛化并且翻译。

2.2.4 开发集，测试集格式处理

NiuTrans 系统进行权重调优的时候需要将开发集处理成一个源语，一个空行，四个目标语的格式作为输入，所以开发集经过预处理分词后，还要进行格式处理。

NiuTrans 系统对测试集结果进行官方评价时，需要生成官方工具需要的 xml 文件，生成工具同样需要将测试集处理成一个源语，一个空行，四个目标语的格式。

脚本：NiuTrans-dev-merge.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-dev-merge.pl \
          源文 \
          译文 1 \
          译文 2 \
          •••
          译文 n   > 标准格式文件

示例：本次基线系统开发集格式处理命令及参数
$> perl   NiuTrans-dev-merge.pl \
          ../data/dev/c.input.txt.token \
          ../data/dev/e.input.ref1.txt.token \
          ../data/dev/e.input.ref2.txt.token \
          ../data/dev/e.input.ref3.txt.token \
          ../data/dev/e.input.ref4.txt.token      >   dev.txt
```

2.2.5 训练集，开发集，测试集，语言模型处理的步骤

本次 Niutrans 基线系统中数据预处理的处理步骤如下:

1 训练集：首先进行双语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。

2 开发集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化

开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行开发集格式处理，处理成一个源语，一个空行和四个目标语的格式。

3 测试集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行测试集格式处理，处理成一个源语，一个空行和四个目标语的格式。

4 语言模型源文件：首先进行单语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。最终使用的语言模型训练文件是处理好的训练集目标语文件与相应的单语语言模型训练文件的合并。以汉英翻译系统为例，语言模型训练文件：处理过的训练集英文部分与处理过的路透社文件的合并，也可以先合并在处理。

注：训练集过滤掉行数请参见附件 1。

2.3 词对齐

本次 NiuTrans 基线系统的训练集词语对齐采用的是开源工具：GIZA++1.0.7。

下载链接：http://code.google.com/p/giza-pp/downloads/detail?name=giza-pp-v1.0.7.tar.gz。

NiuTrans 对该工具进行了封装(用 perl 调用 GIZA++工具)。

训练集词对齐分为两步：首先进行长度比过滤，然后再进行词对齐训练。

训练语料词语对齐的步骤：

1. 对训练语料源语言/目标语言分词结果进行长度比过滤（原文和译文的长度比例限制和最大长度限制）。

长度比过滤脚本：NiuTrans-length.ratio.filter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-length.ratio.filter.pl \
          -src              源文本分词结果 \
          -tgt              目标文本分词结果 \
          -outSrc           源文本分词后长度比过滤结果 \
          -outTgt           目标文本分词后长度比过滤结果 \
          -lowerBoundCoef   源语长度和目标语长度比值下限 \
          -upperBoundCoef   源语长度和目标语长度比值上限
```

示例：部分训练集双语乱码过滤后的文件进行长度比过滤的命令

```
$> perl   NiuTrans-length.ratio.filter.pl \
          -src          ../data/train/NEU/c.input.txt.clean.token \
          -tgt          ../data/train/NEU/e.input.txt.clean.token \
          -outSrc       ../data/train/NEU/c.input.txt.clean.token.filter \
          -outTgt       ../data/train/NEU/e.input.txt.clean.token.filter \
          -lowerBoundCoef 0.4 \
          -upperBoundCoef 1.8
```

注：长度比过滤掉的文件是.discard，过滤掉的行数请参见附件 1。

2. 采用 Niutrans 词对齐脚本跑词对齐（仅支持 Linux 平台）。

词对齐脚本：NiuTrans-running-GIZA++.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                   -src      源文本长度比结果 \
                   -tgt      目标文本长度比结果 \
                   -out      词对齐文件 \
                   -tmpdir   生成临时文件的目录
示例：部分训练集长度比过滤的文件跑词对齐的命令
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                   -src      ../data/train/NEU/c.input.txt.clean.token \
                   -tgt      ../data/train/NEU/e.input.txt.clean.token \
                   -out      ../data/train/NEU/alignment.txt \
                   -tmpdir   ../work/tmp/ &
```

注：由于长度比过滤后的训练集文件较大(总共 450w 左右),跑词对齐时可以分割成 9 个文件，每个 50w 左右，以减少词对齐训练时间。

最后在 train 文件夹里面创建一个 all 文件夹，再把 train 下每个文件夹下的源语、目标语的长度比过滤结果和对应的词对齐文件纵向合并到一起(源语之间合并、目标语之间合并、词对齐之间合并)，分别放到 all 里面(c.input.txt.clean.token ,e.input.txt.clean.token,alignment.txt)，作为训练翻译模型的输入文件。

2.4 生成层次短语规则

层次短语规则是层次短语系统训练得到的短语规则文件，这一步训练会非常慢。生成层次短语规则的输入文件有：训练集双语数据以及相应的词对齐文件。

为了使生成短语规则变得简单，NiuTrans.Hierarchy 系统默认有一个生成层次

短语规则的配置文件。抽层次短语规则时先修改一下配置文件，然后再抽层次短语规则。

抽层次短语规则的配置文件：NiuTrans.hierarchy.train.model.config，在NiuTrans_CWMT\config 目录下。

1．本次汉英层次短语基线系统的修改的配置文件参数，如下图：

```
1  ##############################################
2  ### NiuTrans hierarchy train model config ###
3  ##############################################
4
5  # Hiero Rule table parameters
6  param="srclen"                          value="10"
7  param="tgtlen"                          value="10"
8  param="maxSrcI"                         value="7"
9  param="maxTgtI"                         value="7"
10 param="maxSrcH"                         value="5"
11 param="maxTgtH"                         value="10"
12 param="minSrcSubPhrase"                 value="1"
13 param="minTgtSubPhrase"                 value="1"
14 param="minLexiNum"                      value="1"
15 param="maxnonterm"                      value="2"
16
17 # Hiero Rule table Flag: 1 is true, 0 is false
18 param="alignedLexiReq"                  value="1"
19 param="srcNonTermAdjacent"              value="0"
20 param="tgtNonTermAdjacent"              value="1"
21 param="duplicateHieroRule"              value="0"
22 param="headnonterm"                     value="1"
23 param="null"                            value="0"
24
25 param="printFreq"                       value="0"
26 param="printAlign"                      value="0"
27
28 # Optimization Flag
29 param="unalignedEdgeInit"               value="1"
30 param="unalignedEdgeHiero"              value="0"
31
32 # Optimization Parameters
33 param="maxNulExtSrcInitNum"             value="3"
34 param="maxNulExtTgtInitNum"             value="3"
35 param="maxNulExtSrcHieroNum"            value="1"
36 param="maxNulExtTgtHieroNum"            value="1"
37
38 param="cutoffInit"                      value="0"
39 param="cutoffHiero"                     value="0"
```

value="1"

2．配置文件参数说明：

参数	含义
srclen	源语言端抽取层次短语的初始短语长度
tgtlen	目标语言端抽取层次短语的初始短语长度
maxSrcI	输出的源语言端短语长度
maxTgtI	输出的目标语言端短语长度
maxSrcH	输出的源语言端层次短语长度

maxTgtH	输出的目标语言端层次短语长度
minLexiNum	层次短语中词汇最小个数（建议使用默认值）
maxnonterm	最大泛化槽的个数（建议使用默认值）
srcNonTermAdjacent	源语言端泛化槽相邻开关（建议使用默认值）
tgtNonTermAdjacent	目标语言端泛化槽相邻开关（建议使用默认值）
duplicateHieroRule	一行句对重复输出相同层次短语规则
headnonterm	起始位置为泛化槽开关
null	空翻译规则开关
printFreq	输出（层次）短语对在预料中频次
printAlign	输出（层次）短语对词对齐
unalignedEdgeInit	短语对边界词汇空扩展开关
unalignedEdgeHiero	层次短语规则边界词汇空扩展开关（不建议开启）
maxNulExtSrcInitNum	短语对源语言端空扩展词的个数
maxNulExtTgtInitNum	短语对目标语言端空扩展词的个数
maxNulExtSrcHieroNum	层次短语规则源语言端空扩展词的个数
maxNulExtTgtHieroNum	层次短语规则目标语言端空扩展词的个数
cutoffInit	短语对 cutoff 值（根据频次）
cutoffHiero	层次短语规则 cutoff 值（根据频次）

3．抽层次短语规则的脚本：NiuTrans-hierarchy-train-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-hierarchy-train-model.pl \
          -src      源语言分词文件 \
          -tgt      目标语言分词文件 \
          -aln      源语言到目标语言的词对齐文件
          -out      生成层次短语表
示例: 本次汉英层次短语基线系统抽层次短语规则的命令
$> mkdir ../work/model.hierarchy   -p
$> perl   NiuTrans-hierarchy-train-model.pl \
          -src      ../data/c.input.txt.clean.token \
          -tgt      ../data/e.input.txt.clean.token \
          -aln      ../data/alignment.txt
          -out      ../work/model.hierarchy/hierarchy.rule.table
```

2.5 训练语言模型

本次基线系统的语言模型训练使用的是 NiuTrans 语言模型训练工具。NiuTrans.Hierarchy 汉英基线系统的语言模型源文件是预处理过的路透社英文单语

语料和预处理过的所有训练集英文语料合并到一起。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$>cat    ../data/train/all/e.input.txt.clean.token ../data/LM/router.txt.clean.token >
../data/LM/e.lm.merged.txt
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   语言模型文件 \
          -ngram    n 指 N 元语言模型 \
          -vocab    生成的目标语端语言模型词汇表 \
          -lmbin    生成的目标语端语言模型的二进制文件
示例：本次汉英层次短语基线系统训练语言模型的命令
mkdir ../work/lm/ -p
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   ../data/LM/e.lm.merged.txt \
          -ngram    5 \
          -vocab    ../work/lm/lm.vocab \
          -lmbin    ../work/lm/lm.trie.data \
```

注：汉英层次短语的语言模型和汉英短语系统的语言模型训练是一致的，这里可以直接用汉英短语系统训练好的语言模型。

2.6 生成配置文件

NiuTrans.Hierarchy 将解码所需要用的参数和层次短语表位置写到配置文件里面，解码时直接调用配置文件。

生成配置文件的脚本：Niutrans-hierarchy-generate-mert-config.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl    NiuTrans-hierarchy-generate-mert-config.pl \
          -rule     层次短语规则文件的位置 \
          -lmdir    语言模型文件夹的位置 \
          -nref     参考译文数 \
          -ngram    n(代表 N 元语言模型) \
          -out      生成的配置文件（用于解码）
示例：本次汉英层次短语基线系统生成配置文件的命令
$> mkdir   ../work/config -p
$> perl    NiuTrans-hierarchy-generate-mert-config.pl \
          -rule     ../work/model.hierarchy/hierarchy.rule.table \
          -lmdir    ../work/lm/ \
          -nref     4 \
          -ngram    5 \
```

-out ../work/config/NiuTrans.hierarchy.user.config

解码器配置文件截图：

```
#>>>>>>>>>>>>>>>>>>>>>>
#>>> runtime parameters
# number of MERT iterations
param="nround"                          value="12"
# order of n-gram language model
param="ngram"                           value="5"
# maximum phrase length
param="maxphraselength"                 value="7"
# use punctuation pruning (1) or not (0)
param="usepuncpruning"                  value="1"
# use cube-pruning (1) or not (0)
param="usecubepruning"                  value="1"
# use maxent reordering model (1) or not (0)
param="use-me-reorder"                  value="1"
# use msd reordering model (1) or not (0)
param="use-msd-reorder"                 value="1"
# number of threads
param="nthread"                         value="8"
# how many translations are dumped
param="nbest"                           value="50"
# output OOV words
param="outputoov"                       value="0"
# output source words that are explicitly deleted
param="outputnull"                      value="0"
# beam size (or beam width)
param="beamsize"                        value="50"
# beam scale. This parameter controls the number of hypotheses
# that are computed in decoding.
# e.g., beamscale=3 means we search over 3 * beamsize hyphotheses
param="beamscale"                       value="1"
# number of references of dev. set
param="nref"                            value="4"
# allow null-translation (i.e. word-deletion)
param="usenulltrans"                    value="0"
# allow sequence of null-translations
param="snulltrans"                      value="0"
# normalize output (1) or not (0)
param="normalizeoutput"                 value="0"
# distortion limit
param="maxdd"                           value="15"
```

解码器配置文件参数说明：

参数	含义
Ngram-LanguageModel-File	语言模型二进制文件
Target-Vocab-File	语言模型词汇表
SCFG-Rule-Set	层次短语翻译模型文件
Punct-Vocab-File	解码器标点切分文件（可自定义）
nround	MERT 轮数
ngram	语言模型阶数（最好设置为 5）
maxphraselength	解码加载最大源语言端短语长度
usepuncpruning	解码标点切分开关（加速解码方法）
usecubepruning	cube-pruning 开关（加速解码方法）

nthread	线程数
nbest	生成翻译候选个数
outputoov	输出 OOV 开关
outputnull	输出空翻译开关
beamsize	解码 beam 大小
nref	开发集参考答案个数
usenulltrans	使用空翻译开关
maxdd	最大调序距离
weights	权重（每一维特征使用空格切分）
ranges	开发集特征值调整范围

2.7 规则过滤

由于训练生成的层次短语表非常大，无法全部转载到内存中，并且调优和解码分别是在开发集和测试集上，可以用开发集和测试集合并后来过滤层次短语表。

NiuTrans.Hierarchy 用开发集和测试集合并到一起过滤层次短语表。

规则过滤用到的程序：NiuTrans.PhraseExtractor 。

使用说明：首先进入 NiuTrans_CWMT/目录，然后在命令行下输入：

$> bin/NiuTrans.PhraseExtractor --FILPD \
-dev 开发集和测试集 \
-in 层次短语规则表 \
-out 过滤后的层次短语规则表 \
-maxlen 层次短语规则中规则的最大长度 \
-rnum 开发集中答案的个数

示例：本次汉英层次短语基线系统进行规则过滤的命令

$> cat data/dev/dev.txt data/test/test.txt > data/dev/dev_and_test.txt
$> bin/NiuTrans.PhraseExtractor --FILPD \
-dev data/dev/dev_and_test.txt \
-in work/model.hierarchy/hierarchy.rule.table \
-out work/model.hierarchy/hierarchy.rule.table.filterDevAndTest \
-maxlen 10 \
-rnum 4

注意：过滤完毕后需要将层次短语表的参数改为过滤后的。

$> vim work/config/NiuTrans.hierarchy.user.config \
param = “SCFG-Rule-Set” value=”../work/model.hierarchy/hierarchy.rule.table”
修改为
param=“SCFG-Rule-Set” value=”../work/model.hierarchy/hierarchy.rule.table.filterDevAndTest”

注：dev.txt 是处理好的标准格式(一个源语 N 个目标语)的开发集，test.txt 是处

理成标准格式的测试集。这里测试集可以指定 04 年，05 年的 863 评测数据，07 年 SSMT 数据。

详细说明参见：http://www.nlplab.com/NiuPlan/NiuTrans.Phrase.ch.html 5.规则过滤。

2.8 权重调优

NiuTrans.Hierarchy 汉英基线系统直接在处理好的标准开发集上调优，调优后把最佳权重写到配置文件里面，测试集解码时，直接用开发集上的最优权重。

权重调优脚本：NiuTrans-hierarchy-mert-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-hierarchy-mert-model.pl \
          -config   配置文件\
          -dev      处理好的开发集 \
          -nref     n(参考答案的个数) \
          -round    n(mert 的轮数) \
          -log      生成调优文件的日志
示例：本次汉英短语基线系统权重调优的命令
$> perl   NiuTrans-hierarchy-mert-model.pl \
          -config   ../work/config/NiuTrans.hierarchy.user.config\
          -dev      ../data/dev/dev.txt \
          -nref      4 \
          -round     2 \
          -log      ../work/mert-model.log
```

调优好的 weights：value="0.9187 1.4448 0.5751 0.3996 0.1837 0.2200 -0.6255 0.6500 -0.1200 0.7127 0.3392 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000"。

注：想设置更多权重调优命令，请参见脚本 usage 用法：perl Niutrans-hierarchy-mert-model.pl，然后回车。

2.9 解码

解码这一步在调优之后，直接利用开发集训练好的权重，在测试集上解码。

解码脚本：perl Niutrans-hierarchy-decoder-model.pl 输入：测试集，配置文件，输出：1best.out

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-hierarchy-decoder-model.pl \
          -config      配置文件 \
          -test        预处理好的测试集 \
          -opnull      是否输出 oov(1 是  0 否)
          -output      翻译结果
示例：本次汉英短语基线系统的解码命令以及参数
$> perl   NiuTrans-hierarchy-decoder-model.pl \
          -config      ../work/config/NiuTrans.hierarchy.user.config \
          -test        ../data/test/test.txt \
          -opnull      0 \
          -output      1best.out
```

2.10 恢复大写信息(recasing)

由于测试集是经过预处理分词过的，并且在预处理分词的时候已经把所有大写字母全部转为小写了，所以要对翻译结果恢复大写信息。恢复大写信息需要两步：1 训练大写信息恢复模型(这个和汉英短语系统的模型是一样的，如果汉英短语系统训练的话，这里直接调用就可以)，2 恢复大写信息。

恢复大写信息用到的脚本：NiuTrans-training-recase-model.pl Niutrans-recaser.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-training-recasing-model.pl \
          -corpus      只做 token 的英文语言模型训练文件 \
          -modelDir    生成 recasing 模型路径

$> perl   NiuTrans-recaser.pl \
          -config      恢复大写信息模型的配置文件 \
          -test        翻译结果 \
          -output      恢复大写信息的翻译结果
示例：本次汉英短语基线系统恢复大写信息的命令
$> mkdir   ../work/model.recasing -p
$> perl   NiuTrans-training-recasing-model.pl \
          -corpus      ../data/LM/e.lm.merged.txt \
          -modelDir    ../work/model.recasing/

$> perl   NiuTrans-recaser.pl \
          -config      ../work/config/NiuTrans.phrase.user.config \
          -test        1best.out   \
          -output      1best.out.recased
```

2.11 Detokenizer

这一步对恢复大写的翻译结果进行 detoken 处理。

detoken 脚本：NiuTrans-detokenizer.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  Niutrans-detokenizer.pl \
         -in      恢复大小写的翻译结果 \
         -out     detoken 结果
示例：本次汉英短语基线系统 detoken 命令
$> perl  Niutrans-detokenizer.pl \
         -in      1best.out.recased \
         -out     1best.out.recased.detoken
```

2.12 评价

采用 CWMT2013 官方评价程序，只需将经过后处理的结果变成评价需要的 xml 格式文件，然后输入到官方评价程序即可。这里可以用我们提供 generate-xml-tools（NiuTrans_CWMT/tools/generate-xml-tools）工具生成 xml 文件。generate-xml.pl 是用来生成 xml 的脚本，使用方法如下：

```
$> perl  generate-xml.pl \
         -flist    系统输出的翻译结果文件 \
         -task     汉英新闻翻译默认为“zh_en_news_trans” \
         -1best    \
         -evaluate \
         -rlist      参考答案文件（标准格式）\
         -nref       参考译文数
示例：
$> perl  generate-xml.pl \
         -flist    1best.out.recased.detoken \
         -task     zh_en_news_trans \
         -1best    \
         -evaluate \
         -rlist    dev.C2E.txt \
         -nref      4
```

运行之后生成三个 xml 文件：

- src.xml：源语言 xml 文件
- ref.xml：参考答案 xml 文件

- 1best.xml： 系统输出翻译结果 xml 文件

最后得到的 xml 文件就是评价需要的文件格式。mteval_sbp 是 CWMT 官方提供的评价工具，我们只要把以上三个文件输入到程序中，就可以得到评价结果 1best_result，命令如下：

mteval_sbp -c -r ref.xml -s src.xml -t 1best.xml >1best_result

本次汉英短语基线系统在开发集上调优后，直接带开发集上解码，恢复大小写，detoken，然后进行官方评价的 BLEU_SBP4 值：

大小写不敏感：0.2660(不带 –c 参数)

大小写敏感：0.2503(带 –c 参数)

注：汉英层次短语基线翻译系统的标准配置文件见 NiuTrans_1.3.0_CWMT2013\config\CWMT2013-c2e 目录下：

训练层次短语规则配置文件：NiuTrans.phrase.train.model.CWMT.config

解码器配置文件：NiuTrans.phrase.CWMT.config

3 英汉短语基线翻译系统

整个英汉短语翻译基线系统搭建共分为十一步，这些步骤包括：数据使用，数据预处理，词对齐，训练翻译模型，训练语言模型，生成配置文件，规则过滤，权重调优，解码，后处理，评价。在这些步骤中训练翻译模型和训练语言模型可以并行操作，其他步骤需要按顺序来进行。

3.1 数据使用

评测组织方发放的新闻数据有三种，分别为：训练集、开发集、往年评测数据。其中，训练集是纯文本格式，编码：UTF-8(无 BOM),开发集和往年评测数据是 xml 格式(UTF8)，需要把开发集和往年评测数据从 xml 里面抽出来，处理成一行中文一行英文的纯文本格式 UTF8(无 BOM)。

汉英短语基线系统构建的时候，把往年的评测数据一部分作为测试数据用来评价 NiuTrans.Phrase 的性能，一部分用作训练数据。

训练集是官方提供的训练数据加上 2003 年 863 机器翻译评测数据。

开发集是官方开发集。

测试集是 2004 年 863 机器翻译评测数据，2005 年机器翻译 863 评测数据，2007 年 SSMT 机器翻译评测数据。

3.1.1 训练集

训练集包含以下数据，每个文件夹里只有双语句对，文件格式:UTF8(无 BOM)。存放格式如下：

```
data/train                          #训练集
  |-- Datum                         #点通数据集
  |    |-- c.input.txt              #中文句对
  |    |-- e.input.txt              #英文句对
  |-- HIT-IR
  |-- HIT-MT
  |-- ICT-web-A
  |-- ICT-web-B
  |-- NEU
  |-- XMU-Movie-Subtitle
  |-- CLDC-LAC-2003-006
```

```
|-- CLDC-LAC-2003-004
|-- 2003-863-001
|-- CLDC-LAC-2003-004
|-- 2003-863-001
```

3.1.2 开发集

汉英 1006 句，英汉 1000 句（默认开发集），数据存放格式如下:

```
data/dev                    #开发集
  |--C2E                    #汉英开发集
  |    |-- c.input.txt      #中文句对
  |    |-- e.input.ref1.txt #中文句对翻译 1
  |    |-- e.input.ref2.txt #中文句对翻译 2
  |    |-- e.input.ref3.txt #中文句对翻译 3
  |    |-- e.input.ref4.txt #中文句对翻译 4
  |--E2C
       |-- e.input.txt
       |-- c.input.ref1.txt
       |-- c.input.ref2.txt
       |-- c.input.ref3.txt
       |-- c.input.ref4.txt
```

3.1.3 测试集

选择的测试集有：2004-863-001,2005-863-001,SSMT-2007。从原始 xml 文件抽取出来，保存为 1 个源语 4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下：

```
data/test                          #测试集
  |-- SSMT-2007                    #SSMT-2007 测试集
  |       |-- C2E                  #SSMT-2007 汉英测试集
  |       |       |-- c.input.txt      #中文句对
  |       |       |-- e.input.ref1.txt #中文句对翻译 1
  |       |       |-- e.input.ref2.txt #中文句对翻译 2
  |       |       |-- e.input.ref3.txt #中文句对翻译 3
  |       |       |-- e.input.ref4.txt #中文句对翻译 4
  |       |-- E2C
  |               |-- c.input.txt
  |               |-- e.input.ref1.txt
  |               |-- e.input.ref2.txt
  |               |-- e.input.ref3.txt
  |               |-- e.input.ref4.txt
```

```
|-- 2005-863-001          #同 SSMT-2007
|-- 2004-863-001          #同 SSMT-2007
|-- 2004-863-001          #同 SSMT-2007
```

3.1.4 语言模型训练文件

非评测组织方提供的可用训练数据：路透社语料库和搜狗全网新闻语料库。数据存放格式如下：

```
data/LM                   #语言模型
  |-- sougou.txt          #搜狗中文单语语料
  |-- router.txt          #路透社英文单语语料
```

英汉短语系统最终用到的语言模型训练文件是搜狗全网新闻语料库加上训练集所有中文单语语料。

3.2 数据预处理

数据预处理是统计机器翻译系统的第一步，训练集，开发集，测试集，语言模型文件都需要通过数据预处理。NiuTrans.Phrase 预处理主要包括：乱码过滤，分词，泛化一些数词、时间词、日期词，翻译数词、时间词、日期词等，不同数据处理的步骤也不一样，下面给出本次基线系统数据预处理用到的脚本和不同数据集使用的处理步骤。

为了方便的给出示例，假设所有数据在 NiuTrans_CWMT 目录的 data 文件夹里，里面有文件夹：训练集(train)，存储内容和格式见 1.1 训练集；开发集(dev)，存储内容和格式见 1.2 开发集；测试集(test)，存储内容和格式见 1.3 测试集；语言模型(LM)，包含路透社语料和搜狗语料。

3.2.1 双语乱码过滤

脚本：NiuTrans-clear.illagal.char.pl。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-clear.illegal.char.pl \
         -src      中文文本(无论训练汉英/英汉翻译系统，该参数都指定为中文) \
         -tgt      英文文本(无论训练汉英/英汉翻译系统，该参数都指定为英文) \
         -outSrc   输出乱码过滤后中文文本 \
         -outTgt   输出乱码过滤后英文文本
```

```
示例：部分训练数据进行双语乱码过滤的命令
$> perl   NiuTrans-clear.illegal.char.pl \
          -src      ../data/train/NEU/c.input.txt \
          -tgt      ../data/train/NEU/e.input.txt \
          -outSrc   ../data/train/NEU/c.input.txt.clean \
          -outTgt   ../data/train/NEU/e.input.txt.clean
```

3.2.2 单语乱码过滤

脚本：NiuTrans-monolingual.clear.illegal.char.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      待处理文本 \
          -outTgt   乱码过滤后输出文本
          -lang     语言类别(中文:zh  英文:en)
示例：语言模型(搜狗单语)进行单语乱码过滤的命令
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
          -tgt      ../data/LM/sogou.txt \
          -outTgt   ../data/LM/sogou.txt.clean
          -lang     zh
```

注：双语过滤和单语过滤，过滤掉的乱码存放在 .discard 文件中。

3.2.3 预处理分词

脚本：NiuTrans-running-segmenter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-running-segmenter.pl
          -lang     ch \
          -input    输入待处理文本，这里指乱码过滤后文本 \
          -output   输出预处理分词后文本
          -method   泛化翻译开关(00/01/11)
示例：本次基线系统部分训练数据的分词命令及参数
$> perl   NiuTrans-running-segmenter.pl
          -lang     ch \
          -input    ../data/train/NEU/c.input.txt.clean \
          -output   ../data/train/NEU/c.input.txt.clean.token
          -method   01
```

注：“-lang”为预处理分词文件类型，“ch”表示为中文，“en”表示为英文。

“-method”为预处理分词泛化和翻译功能的开关，00：不泛化也不翻译，01：只泛化不翻译，11：泛化并且翻译。

3.2.4 开发集，测试集格式处理

NiuTrans 系统进行权重调优的时候需要将开发集处理成一个源语，一个空行，四个目标语的格式作为输入，所以开发集经过预处理分词后，还要进行格式处理。

NiuTrans 系统对测试集结果进行官方评价时，需要生成官方工具需要的 xml 文件，生成工具同样需要将测试集处理成一个源语，一个空行，四个目标语的格式。

脚本：NiuTrans-dev-merge.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-dev-merge.pl
          源文 \
          译文 1 \
          译文 2 \
          译文 3 \
          译文 4

示例：本次基线系统开发集格式处理命令及参数
$> perl   NiuTrans-dev-merge.pl
          ../data/dev/c.input.txt.token \
          ../data/dev/e.input.ref1.txt.token \
          ../data/dev/e.input.ref2.txt.token \
          ../data/dev/e.input.ref3.txt.token \
          ../data/dev/e.input.ref4.txt.token
```

3.2.5 训练集，开发集，测试集，语言模型处理的步骤

本次 Niutrans 基线系统中数据预处理的处理步骤如下:

1 训练集：首先进行双语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。

2 开发集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化

开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行开发集格式处理，处理成一个源语，一个空行和四个目标语的格式。

3 测试集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行测试集格式处理，处理成一个源语，一个空行和四个目标语的格式。

4 语言模型源文件：首先进行单语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。最终使用的语言模型训练文件是处理好的训练集目标语文件与相应的单语语言模型训练文件的合并。以汉英翻译系统为例，语言模型训练文件：处理过的训练集英文部分与处理过的路透社文件的合并，也可以先合并在处理。

注：训练集过滤掉行数请参见附件 1。

3.3 词对齐

本次 NiuTrans 基线系统的训练集词语对齐采用的是开源工具：GIZA++1.0.7。

下载链接：

http://code.google.com/p/giza-pp/downloads/detail?name=giza-pp-v1.0.7.tar.gz。

NiuTrans 对该工具进行了封装(用 perl 调用 GIZA++工具)。

训练集词对齐分为两步：首先进行长度比过滤，然后再进行词对齐训练。

训练语料词语对齐的步骤：

1. 对训练语料源语言/目标语言分词结果进行长度比过滤（原文和译文的长度比例限制和最大长度限制）。

长度比过滤脚本：NiuTrans-length.ratio.filter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-length.ratio.filter.pl \
         -src              源文本分词结果 \
         -tgt              目标文本分词结果 \
         -outSrc           源文本分词后长度比过滤结果 \
         -outTgt           目标文本分词后长度比过滤结果 \
         -lowerBoundCoef   源语长度和目标语长度比值下限 \
         -upperBoundCoef   源语长度和目标语长度比值上限
```

```
示例：部分训练集双语乱码过滤后的文件进行长度比过滤的命令
$> perl   NiuTrans-length.ratio.filter.pl \
          -src              ../data/train/NEU/c.input.txt.clean.token \
          -tgt              ../data/train/NEU/e.input.txt.clean.token \
          -outSrc           ../data/train/NEU/c.input.txt.clean.token.filter \
          -outTgt           ../data/train/NEU/e.input.txt.clean.token.filter \
          -lowerBoundCoef 0.4 \
          -upperBoundCoef 1.8
```

注：长度比过滤掉的文件是.discard，过滤掉的行数请参见附件 1。

2. 采用 Niutrans 词对齐脚本跑词对齐（仅支持 Linux 平台）。

词对齐脚本：NiuTrans-running-GIZA++.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                   -src      源文本长度比结果 \
                   -tgt      目标文本长度比结果 \
                   -out      词对齐文件 \
                   -tmpdir   生成临时文件的目录
示例：部分训练集长度比过滤的文件跑词对齐的命令
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                   -src      ../data/train/NEU/c.input.txt.clean.token \
                   -tgt      ../data/train/NEU/e.input.txt.clean.token \
                   -out      ../data/train/NEU/alignment.txt \
                   -tmpdir   ../work/tmp/ &
```

注：由于长度比过滤后的训练集文件较大(总共 450w 左右),跑词对齐时可以分割成 9 个文件，每个 50w 左右，以减少词对齐训练时间。

由于在汉英任务时已经得到中文为源语言、英文为目标语言的词对齐文件，所以我们实际上只需要把中-英词对齐文件转化成英-中词对齐文件即可，这里我们采用 NiuTrans-convert.alignment.to.inv.pl 脚本。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
perl NiuTrans-convert.alignment.to.inv.pl  中-英词对齐文件  英-中词对齐文件

示例：
perl NiuTrans-convert.alignment.to.inv.pl
    ../data/train/NEU/alignment.txt
    ../data/train/NEU/alignment_e2c.txt
```

最后在 train 文件夹里面创建一个 all 文件夹，再把 train 下每个文件夹下的源语、目标语的长度比过滤结果和对应的词对齐文件纵向合并到一起(源语之间合并、目标语之间合并、词对齐之间合并)，分别放到 all 里面(c.input.txt.clean.token ,e.input.txt.clean.token,alignment.txt)，作为训练翻译模型的输入文件。

3.4 训练翻译模型

预处理完训练集并且有了词语对齐之后，就可以训练翻译模型了。为了使训练翻译模型变得简单，NiuTrans.Phrase 系统默认有一个训练翻译模型的配置文件。先修改一下配置文件，然后训练翻译模型。

训练翻译模型的配置文件：NiuTrans.phrase.train.model.config，在 NiuTrans_CWMT\config 目录下。

1．本次 Baseline 英汉翻译系统的配置，如下图：

```
########################################
### NiuTrans  phrase train model config ###
########################################

# temp file path
param="Lexical-Table"                value="lex"
param="Extract-Phrase-Pairs"         value="extractPhrasePairs"

# phrase table parameters
param="Max-Source-Phrase-Size"       value="3"
param="Max-Target-Phrase-Size"       value="5"
param="Phrase-Cut-Off"               value="0"

# phrase translation model
param="Phrase-Table"                 value="phrase.translation.table"

# maxent lexicalized reordering parameters
param="ME-max-src-phrase-len"        value="3"
param="ME-max-tar-phrase-len"        value="5"
param="ME-null-algn-word-num"        value="1"
param="ME-max-sample-num"            value="15000000"
param="ME-use-src-parse-pruning"     value="0"
param="ME-src-parse-path"            value="/path/to/parse.tree"

# maxent lexicalized reordering model
param="ME-Reordering-Table"          value="me.reordering.table"

# msd lexicalized reordering parameters
param="MSD-model-type"               value="1"

# msd lexicalized reordering model
param="MSD-Reordering-Model"         value="msd.reordering.table"
```

注：以上参数就是本次 baseline 系统采用的参数设置。其中

Max-Source-Phrase-Size 值为 3

Max-Target-Phrase-Size 值为 5

ME-max-src-phrase-len 值为 3

ME-max-tar-phrase-len 值为 5

ME-max-sample-num 值为 15000000

2．配置文件参数说明：

参数	含义
Lexical-Table	词汇翻译表（临时文件）
Extract-Phrase-Pairs	短语对集合文件（临时文件）
Max-Source-Phrase-Size	短语对源语言端短语长度
Max-Target-Phrase-Size	短语对目标语言端短语长度
Phrase-Table	短语翻译表
ME-max-src-phrase-len	ME 调序模型源语言端短语长度
ME-max-tar-phrase-len	ME 调序模型目标语言端短语长度
ME-max-sample-num	ME 模型训练抽取 sample 个数
ME-Reordering-Table	ME 调序模型
MSD-model-type	MSD 调序模型类型
MSD-Reordering-Model	MSD 调序模型

3．训练翻译模型的脚本：NiuTrans-phrase-train-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

**perl NiuTrans-phrase-train-model.pl **
-tmdir 生成翻译规则表的目录
-s 源语言分词文件
-t 目标语言分词文件
-a 源语言到目标语言的词对齐文件
示例: 假设所有训练数据已经合并到 train 目录下。
mkdir ../work/model -p
**perl NiuTrans-phrase-train-model.pl **
-tmdir ../work/model/ \
-s ../data/train/e.input.txt.clean.token \
-t ../data/train/c.input.txt.clean.token \
-a ../data/train/alignment.txt

详细说明参见 http://www.nlplab.com/NiuPlan/NiuTrans.Phrase.ch.html 2.训练翻译模型和高级设置。

注：训练 ME 调序模型需要较大的内存，如果程序运行过程中出现异常，建议减小 ME-max-sample-num 参数值。

3.5 训练语言模型

```
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   语言模型文件 \
          -ngram    n 指 N 元语言模型 \
          -vocab    生成的目标语端语言模型词汇表 \
          -lmbin    生成的目标语端语言模型的二进制文件
示例：以英汉为例，汉语训练数据已和搜狗分词语料合并为 c.lm.merged.txt
mkdir ../work/lm/ -p
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   ../data/LM/c.lm.merged.txt \
          -ngram    5 \
          -vocab    ../work/lm/lm.vocab \
          -lmbin    ../work/lm/lm.trie.data \
```

语言模型训练使用 NiuTrans.Phrase 语言模型训练工具。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入上图中命令：

语言模型训练时采用-ngram 参数为 5。

3.6 生成配置文件

NiuTrans.Phrase 将解码需要用的参数和翻译模型，调序模型写到配置文件里面。

生成配置文件的脚本：Niutrans-phrase-generate-mert-config.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl    NiuTrans-phrase-generate-mert-config.pl \
           -tmdir    翻译模型文件夹的位置 \
           -lmdir    语言模型文件夹的位置 \
           -ngram    n(代表 N 元语言模型) \
           -o        生成的配置文件（用于解码）
```

示例：

```
$> mkdir   ../work/config -p
$> perl    NiuTrans-phrase-generate-mert-config.pl \
           -tmdir    ../work/model/ \
           -lmdir    ../work/lm/ \
           -ngram    5 \
           -o        ../work/config/NiuTrans.phrase.user.config
```

解码器配置文件截图：

```
#>>> runtime parameters
# number of MERT iterations
param="nround"                          value="12"
# order of n-gram language model
param="ngram"                           value="5"
# maximum phrase length
param="maxphraselength"                 value="7"
# use punctuation pruning (1) or not (0)
param="usepuncpruning"                  value="1"
# use cube-pruning (1) or not (0)
param="usecubepruning"                  value="1"
# use maxent reordering model (1) or not (0)
param="use-me-reorder"                  value="1"
# use msd reordering model (1) or not (0)
param="use-msd-reorder"                 value="1"
# number of threads
param="nthread"                         value="8"
# how many translations are dumped
param="nbest"                           value="50"
# output OOV words
param="outputoov"                       value="0"
# output source words that are explicitly deleted
param="outputnull"                      value="0"
# beam size (or beam width)
param="beamsize"                        value="50"
# beam scale. This parameter controls the number of hypotheses
# that are computed in decoding.
# e.g., beamscale=3 means we search over 3 * beamsize hyphotheses
param="beamscale"                       value="1"
# number of references of dev. set
param="nref"                            value="4"
# allow null-translation (i.e. word-deletion)
param="usenulltrans"                    value="0"
# allow sequence of null-translations
param="snulltrans"                      value="1"
# normalize output (1) or not (0)
param="normalizeoutput"                 value="0"
# distortion limit
param="maxdd"                           value="8"
# lowercase translation result
param="lowertext"                       value="1"
```

解码器配置文件参数说明：

参数	含义
Ngram-LanguageModel-File	语言模型二进制文件
Target-Vocab-File	语言模型词汇表
ME-Reordering-Table	ME 调序模型文件
MSD-Reordering-Model	MSD 调序模型文件
Phrase-Table	短语翻译表文件
Punct-Vocab-File	解码器标点切分文件（可自定义）
nround	MERT 轮数
ngram	语言模型阶数（最好设置为 5）
usepuncpruning	解码标点切分开关（加速解码方法）
usecubepruning	cube-pruning 开关（加速解码方法）
use-me-reorder	使用 ME 模型开关
use-msd-reorder	使用 MSD 模型开关
nthread	线程数
nbest	生成翻译候选个数
outputoov	输出 OOV 开关
outputnull	输出空翻译开关
beamsize	解码 beam 大小
nref	开发集参考答案个数
usenulltrans	使用空翻译开关
maxdd	最大调序距离
lowertext	解码过程英文小写化
weights	权重（每一维特征使用空格切分）
ranges	开发集特征值调整范围

3.7 规则过滤

由于训练生成翻译模型和调序模型比较大，无法全部转载到内存中，因为调优和解码是在开发集和测试集上，可以用开发集和测试集过滤翻译模型和调序模型。NiuTrans.Phrase baseline 用开发集和测试集合并到一起过滤翻译模型和 MSD 调序模型。

规则过滤用到的程序：NiuTrans.PhraseExtractor，脚本:filter.msd.model.pl

使用说明：首先进入 NiuTrans_CWMT/目录，然后在命令行下输入：

```
$> bin/NiuTrans.PhraseExtractor --FILPD \
        -dev      开发集和测试集 \
```

-in 短语翻译表 \
-out 过滤后的短语翻译表 \
-maxlen 翻译表中的最大长度 \
-rnum 开发集中答案的个数

$> perl scripts/filter.msd.model.pl \
过滤后的短语翻译表
MSD 调序表 > 过滤后的 MSD 调序表

示例：

$> cat data/dev/dev.txt data/test/test.txt > data/dev/dev_and_test.txt

$> bin/NiuTrans.PhraseExtractor --FILPD \
-dev data/dev/dev_and_test.txt \
-in work/model/phrase.translation.table \
-out work/model/phrase.translation.table.filterDevAndTest \
-maxlen 10 \
-rnum 4

$> perl scripts/filter.msd.model.pl \
work/model/phrase.translation.table.filterDevAndTest \
work/model/msd.reordering.table > work/model/msd.reordering.table.filter DevAndTest

注意：规则过滤后需要将翻译表和 MSD 调序表的参数替换为过滤后的。

$> vim ../work/config/NiuTrans.phrase.user.config

param = “MSD-Reordering-Model” value = “../work/model.phrase/msd.reordering.table”

改为

param = “MSD-Reordering-Model” value = “../work/model.phrase/msd.reordering.table. filterDevAndTest”

param = “Phrase-Table” value = “../work/model.phrase/phrase.translation.table”

改为

param = “Phrase-Table” value = “../work/model.phrase/phrase.translation.table. filterDevAndTest”

param="nbest"设为 value="50"

param="beamsize"设为 value="50"

本次 baseline 系统中-maxlen 参数设为 10，-rnum 参数设为 4。

详细说明参见：http://www.nlplab.com/NiuPlan/NiuTrans.Phrase.ch.html 5.规则过滤。

3.8 权重调优

NiuTrans.Phrase baseline 直接在处理好的标准开发集上调优，调优后把最佳权

重写到配置文件里面，解码的时候直用最优权重。

权重调优脚本：NiuTrans-phrase-mert-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-phrase-mert-model.pl \
         -dev    处理好的开发集(1 个汉语一个空行 4 个目标语的格式) \
         -c      配置文件 \
         -nref   n(参考答案的个数) \
         -l      生成调优文件的日志
示例：
$> perl  NiuTrans-phrase-mert-model.pl \
         -dev    ../data/dev/dev.txt \
         -c      ../work/config/NiuTrans.phrase.user.config \
         -nref   4 \
         -l      ../work/mert-model.log
```

注：想设置更多权重调优命令，请参见脚本 usage 用法：perl Niutrans-phrase-mert-model.pl 回车。

本次英汉短语 baseline 系统最后使用的参数如下：

param="weights" value=" 2.0080 1.6689 0.9979 0.1577 0.5808 0.3900 0.9585 0.9100 -0.0100 2.6972 0.0000 0.4922 0.0258 0.3000 0.0803 0.0848 0.3000"

3.9 解码

解码这一步在调优之后，直接利用开发集训练好的权重，在测试集上解码。

解码脚本：perl Niutrans-phrase-decoder-model.pl 输入：测试集，配置文件，输出：1best.out

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-phrase-decoder-model.pl \
         -test   测试集 \
         -c      配置文件 \
         -output 翻译结果 \
         -opnull 是否输出 oov（默认为 1 输出，0 为不输出）
示例：
$> perl  NiuTrans-phrase-decoder-model.pl \
         -test   ../data/test/all_test.txt \
         -c      ../work/config/NiuTrans.phrase.user.config \
         -output 1best.out
         -opnull 0
```

3.10 后处理

Baseline 解码出来的结果在评价之前需要先进行后处理。系统最终的输出结果是分词的，而且中文标点符号变为英文的符号，对应关系如下：

系统输出翻译结果	中文标点
. . .	…
.	。
<<	《
>>	》
“”	“ ”

在后处理过程中我们会做以下处理：

1. 首先把系统输出结果中文之间的空格去掉。
2. 按照上面的表格，把系统输出结果的标点转换成对应的中文标点。以上两步我们可以通过系统提供的脚本（NiuTrans_CWMT/tools/generate-xml-tools/punc.postedit.pl）进行处理。
3. 按照往年的评价规范，我们把系统的输出结果进行 A3 区的全角到半角的转换，这个步骤可以用我们提供的“英汉 A3 区全半角转换工具”（NiuTrans_CWMT/tools/en2ch.A3）完成，使用方法见 readme.txt。

另外需要注意的一点是，汉语中的句号、书名号、双引号和顿号为全角，无法替换成半角，所以这部分符号在进行 A3 区全半角转换的时候没有进行改动。

3.11 评价

采用 CWMT2013 官方评价程序，只需将经过后处理的结果变成评价需要的 xml 格式文件，然后输入到官方评价程序即可。我们搭建系统的时候实际上是先进行上一步后处理的前两步，即先把系统输出结果中文之间的空格去掉、标点转换，然后生成 xml 文件。这里可以用我们提供 generate-xml-tools（NiuTrans_CWMT/tools/generate-xml-tools）工具，它可以完成后处理前两步，并生成 xml 文件。generate-xml.pl 是用来生成 xml 的脚本，使用方法如下：

```
$> perl   generate-xml.pl \
          -flist      系统输出的翻译结果文件 \
          -task       英汉新闻翻译默认为“en_zh_news_trans” \
          -1best      \
          -evaluate \
```

```
        -rlist      参考答案文件\
        -nref       参考译文数
示例：
$> perl generate-xml.pl \
        -flist      1best.dev.out \
        -task       en_zh_news_trans \
        -1best      \
        -evaluate \
        -rlist      ref.dev.E2C.txt \
        -nref       4
```

运行之后生成三个 xml 文件：

- src.xml：源语言 xml 文件
- ref.xml：参考答案 xml 文件
- 1best.xml： 系统输出翻译结果 xml 文件

如果出现结果文件与标准答案标点符号不统一的情况，则可用 A3 区全半角转换工具进行转换统一标点符号。

最后得到的 xml 文件就是评价需要的文件格式。mteval_sbp 是 CWMT 官方提供的评价工具，我们只要把以上三个文件输入到程序中，就可以得到评价结果 1best_result，命令如下：

mteval_sbp -c -r ref.xml -s src.xml -t 1best.xml >1best_result

本次英汉层次短语基线系统在 dev 上的 BLEU_SBP5 值：0.3256。

注：英汉短语基线翻译系统的标准配置文件见 NiuTrans_1.3.0_CWMT2013\config\CWMT2013-e2c 目录下：

训练翻译模型的配置文件：NiuTrans.phrase.train.model.CWMT.config

解码器配置文件：NiuTrans.phrase.CWMT.config

4 英汉层次短语基线翻译系统

整个英汉层次短语翻译基线系统搭建共分为十一步，这些步骤包括：数据使用，数据预处理，词对齐，生成层次短语规则，训练语言模型，生成配置文件，规则过滤，权重调优，解码，后处理，评价。在这些步骤中训练翻译模型和训练语言模型可以并行操作，其他步骤需要按顺序来进行。

4.1 数据使用

评测组织方发放的新闻数据有三种，分别为：训练集、开发集、往年评测数据。其中，训练集是纯文本格式，编码：UTF-8(无 BOM),开发集和往年评测数据是 xml 格式(UTF8)，需要把开发集和往年评测数据从 xml 里面抽出来，处理成一行中文一行英文的纯文本格式 UTF8(无 BOM)。

汉英短语基线系统构建的时候，把往年的评测数据一部分作为测试数据用来评价 NiuTrans. Hierarchy 的性能，一部分用作训练数据。

训练集是官方提供的训练数据加上 2003 年 863 机器翻译评测数据。

开发集是官方开发集。

测试集是 2004 年 863 机器翻译评测数据，2005 年机器翻译 863 评测数据，2007 年 SSMT 机器翻译评测数据。

4.1.1 训练集

训练集包含以下数据，每个文件夹里只有双语句对，文件格式:UTF8(无 BOM)。存放格式如下：

```
data/train                          #训练集
  |-- Datum                         #点通数据集
  |    |-- c.input.txt              #中文句对
  |    |-- e.input.txt              #英文句对
  |-- HIT-IR
  |-- HIT-MT
  |-- ICT-web-A
  |-- ICT-web-B
  |-- NEU
  |-- XMU-Movie-Subtitle
```

```
|-- CLDC-LAC-2003-006
|-- CLDC-LAC-2003-004
|--2003-863-001
|-- CLDC-LAC-2003-004
|--2003-863-001
```

4.1.2 开发集

汉英 1006 句，英汉 1000 句（默认开发集），数据存放格式如下:

```
data/dev                    #开发集
  |--C2E                    #汉英开发集
  |    |-- c.input.txt      #中文句对
  |    |-- e.input.ref1.txt #中文句对翻译 1
  |    |-- e.input.ref2.txt #中文句对翻译 2
  |    |-- e.input.ref3.txt #中文句对翻译 3
  |    |-- e.input.ref4.txt #中文句对翻译 4
  |--E2C
       |-- e.input.txt
       |-- c.input.ref1.txt
       |-- c.input.ref2.txt
       |-- c.input.ref3.txt
       |-- c.input.ref4.txt
```

4.1.3 测试集

选择的测试集有：2004-863-001,2005-863-001,SSMT-2007。从原始 xml 文件抽取出来，保存为 1 个源语 4 个对应翻译的纯文本文件(UTF8 无 BOM)。数据存放格式如下：

```
data/test                          #测试集
  |-- SSMT-2007                    #SSMT-2007 测试集
  |       |-- C2E                  #SSMT-2007 汉英测试集
  |       |       |-- c.input.txt      #中文句对
  |       |       |-- e.input.ref1.txt #中文句对翻译 1
  |       |       |-- e.input.ref2.txt #中文句对翻译 2
  |       |       |-- e.input.ref3.txt #中文句对翻译 3
  |       |       |-- e.input.ref4.txt #中文句对翻译 4
  |       |-- E2C
  |               |-- c.input.txt
  |               |-- e.input.ref1.txt
  |               |-- e.input.ref2.txt
  |               |-- e.input.ref3.txt
```

```
|              |-- e.input.ref4.txt
|-- 2005-863-001                #同 SSMT-2007
|-- 2004-863-001                #同 SSMT-2007
|-- 2004-863-001                #同 SSMT-2007
```

4.1.4 语言模型训练文件

非评测组织方提供的可用训练数据：路透社语料库和搜狗全网新闻语料库。数据存放格式如下：

```
data/LM                       #语言模型
  |-- sougou.txt              #搜狗中文单语语料
  |-- router.txt              #路透社英文单语语料
```

英汉短语系统最终用到的语言模型训练文件是搜狗全网新闻语料库加上训练集所有中文单语语料。

4.2 数据预处理

数据预处理是统计机器翻译系统的第一步，训练集，开发集，测试集，语言模型文件都需要通过数据预处理。NiuTrans.Phrase 预处理主要包括：乱码过滤，分词，泛化一些数词、时间词、日期词，翻译数词、时间词、日期词等，不同数据处理的步骤也不一样，下面给出本次基线系统数据预处理用到的脚本和不同数据集使用的处理步骤。

为了方便的给出示例，假设所有数据在 NiuTrans_CWMT 目录的 data 文件夹里，里面有文件夹：训练集(train)，存储内容和格式见 1.1 训练集；开发集(dev)，存储内容和格式见 1.2 开发集；测试集(test)，存储内容和格式见 1.3 测试集；语言模型(LM)，包含路透社语料和搜狗语料。

4.2.1 双语乱码过滤

脚本：NiuTrans-clear.illagal.char.pl。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl  NiuTrans-clear.illegal.char.pl \
         -src      中文文本(无论训练汉英/英汉翻译系统，该参数都指定为中文) \
         -tgt      英文文本(无论训练汉英/英汉翻译系统，该参数都指定为英文) \
         -outSrc   输出乱码过滤后中文文本 \
```

```
        -outTgt   输出乱码过滤后英文文本

示例：部分训练数据进行双语乱码过滤的命令
$> perl   NiuTrans-clear.illegal.char.pl \
        -src      ../data/train/NEU/c.input.txt \
        -tgt      ../data/train/NEU/e.input.txt \
        -outSrc   ../data/train/NEU/c.input.txt.clean \
        -outTgt   ../data/train/NEU/e.input.txt.clean
```

4.2.2 单语乱码过滤

脚本：NiuTrans-monolingual.clear.illegal.char.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
        -tgt      待处理文本 \
        -outTgt   乱码过滤后输出文本
        -lang     语言类别(中文:zh  英文:en)
示例：语言模型(搜狗单语)进行单语乱码过滤的命令
$> perl   NiuTrans-monolingual.clear.illegal.char.pl \
        -tgt      ../data/LM/sogou.txt \
        -outTgt   ../data/LM/sogou.txt.clean
        -lang     zh
```

注：双语过滤和单语过滤，过滤掉的乱码存放在 .discard 文件中。

4.2.3 预处理分词

脚本：NiuTrans-running-segmenter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-running-segmenter.pl
        -lang     ch \
        -input    输入待处理文本，这里指乱码过滤后文本 \
        -output   输出预处理分词后文本
        -method   泛化翻译开关(00/01/11)
示例：本次基线系统部分训练数据的分词命令及参数
$> perl   NiuTrans-running-segmenter.pl
        -lang     ch \
        -input    ../data/train/NEU/c.input.txt.clean \
        -output   ../data/train/NEU/c.input.txt.clean.token
```

```
        -method    01
```

注：“-lang”为预处理分词文件类型，“ch”表示为中文，“en”表示为英文。

“-method”为预处理分词泛化和翻译功能的开关，00：不泛化也不翻译，01：只泛化不翻译，11：泛化并且翻译。

4.2.4 开发集，测试集格式处理

NiuTrans 系统进行权重调优的时候需要将开发集处理成一个源语，一个空行，四个目标语的格式作为输入，所以开发集经过预处理分词后，还要进行格式处理。

NiuTrans 系统对测试集结果进行官方评价时，需要生成官方工具需要的 xml 文件，生成工具同样需要将测试集处理成一个源语，一个空行，四个目标语的格式。

脚本：NiuTrans-dev-merge.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl    NiuTrans-dev-merge.pl
           源文 \
           译文 1 \
           译文 2 \
           译文 3 \
           译文 4

示例：本次基线系统开发集格式处理命令及参数
$> perl    NiuTrans-dev-merge.pl
           ../data/dev/c.input.txt.token \
           ../data/dev/e.input.ref1.txt.token \
           ../data/dev/e.input.ref2.txt.token \
           ../data/dev/e.input.ref3.txt.token \
           ../data/dev/e.input.ref4.txt.token
```

4.2.5 训练集，开发集，测试集，语言模型处理的步骤

本次 Niutrans 基线系统中数据预处理的处理步骤如下:

1 训练集：首先进行双语乱码过滤，然后对过滤结果分别预处理分词，开启泛

化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。

2 开发集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行开发集格式处理，处理成一个源语，一个空行和四个目标语的格式。

3 测试集：分别对源语和目标语进行预处理分词。源语分词时，同时开启泛化开关和翻译开关，泛化并翻译数词，时间词，日期词；目标语关闭泛化开关和翻译开关；最后进行测试集格式处理，处理成一个源语，一个空行和四个目标语的格式。

4 语言模型源文件：首先进行单语乱码过滤，然后对过滤结果分别预处理分词，开启泛化开关，泛化数词($number)、时间词($time)、日期词($date)，关闭翻译开关。最终使用的语言模型训练文件是处理好的训练集目标语文件与相应的单语语言模型训练文件的合并。以汉英翻译系统为例，语言模型训练文件：处理过的训练集英文部分与处理过的路透社文件的合并，也可以先合并在处理。

注：训练集过滤掉行数请参见附件 1。

4.3 词对齐

本次 NiuTrans 基线系统的训练集词语对齐采用的是开源工具：GIZA++1.0.7。

下载链接：http://code.google.com/p/giza-pp/downloads/detail?name=giza-pp-v1.0.7.tar.gz。

NiuTrans 对该工具进行了封装(用 perl 调用 GIZA++工具)。

训练集词对齐分为两步：首先进行长度比过滤，然后再进行词对齐训练。

训练语料词语对齐的步骤：

1.对训练语料源语言/目标语言分词结果进行长度比过滤（原文和译文的长度比例限制和最大长度限制）。

长度比过滤脚本：NiuTrans-length.ratio.filter.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl NiuTrans-length.ratio.filter.pl \
        -src            源文本分词结果 \
        -tgt            目标文本分词结果 \
        -outSrc         源文本分词后长度比过滤结果 \
        -outTgt         目标文本分词后长度比过滤结果 \
        -lowerBoundCoef 源语长度和目标语长度比值下限 \
```

```
        -upperBoundCoef 源语长度和目标语长度比值上限
示例：部分训练集双语乱码过滤后的文件进行长度比过滤的命令
$> perl  NiuTrans-length.ratio.filter.pl \
        -src            ../data/train/NEU/c.input.txt.clean.token \
        -tgt            ../data/train/NEU/e.input.txt.clean.token \
        -outSrc         ../data/train/NEU/c.input.txt.clean.token.filter \
        -outTgt         ../data/train/NEU/e.input.txt.clean.token.filter \
        -lowerBoundCoef 0.4 \
        -upperBoundCoef 1.8
```

注：长度比过滤掉的文件是.discard，过滤掉的行数请参见附件 1。

2.采用 Niutrans 词对齐脚本跑词对齐（仅支持 Linux 平台）。

词对齐脚本：NiuTrans-running-GIZA++.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                -src    源文本长度比结果 \
                -tgt    目标文本长度比结果 \
                -out    词对齐文件 \
                -tmpdir 生成临时文件的目录
示例：部分训练集长度比过滤的文件跑词对齐的命令
$> nohup nice perl NiuTrans-running-GIZA++.pl \
                -src    ../data/train/NEU/c.input.txt.clean.token \
                -tgt    ../data/train/NEU/e.input.txt.clean.token \
                -out    ../data/train/NEU/alignment.txt \
                -tmpdir ../work/tmp/ &
```

注：由于长度比过滤后的训练集文件较大(总共 450w 左右),跑词对齐时可以分割成 9 个文件，每个 50w 左右，以减少词对齐训练时间。

由于在汉英任务时已经得到中文为源语言、英文为目标语言的词对齐文件，所以我们实际上只需要把中-英词对齐文件转化成英-中词对齐文件即可，这里我们采用 NiuTrans-convert.alignment.to.inv.pl 脚本。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
perl NiuTrans-convert.alignment.to.inv.pl 中-英词对齐文件 英-中词对齐文件

示例：
perl NiuTrans-convert.alignment.to.inv.pl
    ../data/train/NEU/alignment.txt
```

../data/train/NEU/alignment_e2c.txt

最后在 train 文件夹里面创建一个 all 文件夹，再把 train 下每个文件夹下的源语、目标语的长度比过滤结果和对应的词对齐文件纵向合并到一起(源语之间合并、目标语之间合并、词对齐之间合并)，分别放到 all 里面(c.input.txt.clean.token ,e.input.txt.clean.token,alignment.txt)，作为训练翻译模型的输入文件。

4.4 生成层次短语规则

```
###############################################
### NiuTrans hierarchy train model config ###
###############################################

# Hiero Rule table parameters
param="srclen"                          value="10"
param="tgtlen"                          value="10"
param="maxSrcI"                         value="7"
param="maxTgtI"                         value="7"
param="maxSrcH"                         value="5"
param="maxTgtH"                         value="10"
param="minSrcSubPhrase"                 value="1"
param="minTgtSubPhrase"                 value="1"
param="minLexiNum"                      value="1"
param="maxnonterm"                      value="2"

# Hiero Rule table Flag: 1 is true, 0 is false
param="alignedLexiReq"                  value="1"
param="srcNonTermAdjacent"              value="0"
param="tgtNonTermAdjacent"              value="1"
param="duplicateHieroRule"              value="0"
param="headnonterm"                     value="1"
param="null"                            value="0"

param="printFreq"                       value="0"
param="printAlign"                      value="0"

# Optimization Flag
param="unalignedEdgeInit"               value="1"
param="unalignedEdgeHiero"              value="0"

# Optimization Parameters
param="maxNulExtSrcInitNum"             value="3"
param="maxNulExtTgtInitNum"             value="3"
param="maxNulExtSrcHieroNum"            value="1"
param="maxNulExtTgtHieroNum"            value="1"

param="cutoffInit"                      value="0"
param="cutoffHiero"                     value="0"
```

注：以上参数就是本次 baseline 系统采用的参数设置。

配置文件参数说明：

参数	含义
srclen	源语言端抽取层次短语的初始短语长度
tgtlen	目标语言端抽取层次短语的初始短语长度
maxSrcI	输出的源语言端短语长度
maxTgtI	输出的目标语言端短语长度
maxSrcH	输出的源语言端层次短语长度
maxTgtH	输出的目标语言端层次短语长度
minLexiNum	层次短语中词汇最小个数（建议使用默认值）
maxnonterm	最大泛化槽的个数（建议使用默认值）
srcNonTermAdjacent	源语言端泛化槽相邻开关（建议使用默认值）
tgtNonTermAdjacent	目标语言端泛化槽相邻开关（建议使用默认值）
duplicateHieroRule	一行句对重复输出相同层次短语规则
headnonterm	起始位置为泛化槽开关
null	空翻译规则开关
printFreq	输出（层次）短语对在预料中频次
printAlign	输出（层次）短语对词对齐
unalignedEdgeInit	短语对边界词汇空扩展开关
unalignedEdgeHiero	层次短语规则边界词汇空扩展开关（不建议开启）
maxNulExtSrcInitNum	短语对源语言端空扩展词的个数
maxNulExtTgtInitNum	短语对目标语言端空扩展词的个数
maxNulExtSrcHieroNum	层次短语规则源语言端空扩展词的个数
maxNulExtTgtHieroNum	层次短语规则目标语言端空扩展词的个数
cutoffInit	短语对 cutoff 值（根据频次）
cutoffHiero	层次短语规则 cutoff 值（根据频次）

层次短语规则是层次短语系统训练得到的短语规则文件，生成层次短语规则的输入文件有：源语言分词文件，目标语言分词文件，源语言到目标语言的词对齐文件。

为了使生成短语规则变得简单，NiuTrans.Hierarchy 系统默认有一个生成层次短语规则的配置文件。抽层次短语规则时先修改一下配置文件，然后在生成层次短语规则。

抽层次短语规则的配置文件：NiuTrans.hierarchy.train.model.config，在 NiuTrans_CWMT\config 目录下。

1．本次基线系统的配置文件，如上图所示。

2．训练短语规则的脚本：NiuTrans-hierarchy-train-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-hierarchy-train-model.pl \
          -src      源语言分词文件 \
          -tgt      目标语言分词文件 \
          -aln      源语言到目标语言的词对齐文件
          -out      生成层次短语表
示例: 假设所有训练数据已经合并到 train 目录下。
$> mkdir ../work/model.hierarchy   -p
$> perl   NiuTrans-hierarchy-train-model.pl \
          -src      ../data/e.input.txt.clean.token.filter \
          -tgt      ../data/c.input.txt.clean.token.filter \
          -aln      ../data/alignment.txt
          -out      ../work/model.hierarchy/hierarchy.rule.table
```

详细说明参见 http://www.nlplab.com/NiuPlan/NiuTrans.Hierarchy.ch.html 2.训练翻译模型和高级设置。

4.5 训练语言模型

语言模型训练使用层次短语系统语言模型训练工具。

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   语言模型文件 \
          -ngram    n 指 N 元语言模型 \
          -vocab    生成的目标语端语言模型词汇表 \
          -lmbin    生成的目标语端语言模型的二进制文件
示例：以英汉为例，汉语训练数据已和搜狗分词语料合并为 c.lm.merged.txt
mkdir ../work/lm/ -p
$> perl   NiuTrans-training-ngram-LM.pl \
          -corpus   ../data/LM/c.lm.merged.txt \
          -ngram    5 \
          -vocab    ../work/lm/lm.vocab \
          -lmbin    ../work/lm/lm.trie.data
```

语言模型训练时采用-ngram 参数为 5。

4.6 生成配置文件

NiuTrans.Hierarchy 将解码所需要用的参数和层次短语表位置写到配置文件里面，解码时直接调用配置文件。

生成配置文件的脚本：NiuTrans-hierarchy-generate-mert-config.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

$> perl NiuTrans-hierarchy-generate-mert-config.pl \
-rule 层次短语规则文件的位置 \
-lmdir 语言模型文件夹的位置 \
-nref 参考译文数 \
-ngram n(代表 N 元语言模型) \
-out 生成的配置文件（用于解码）

示例：

$> mkdir ../work/config -p
$> perl NiuTrans-hierarchy-generate-mert-config.pl \
-rule ../work/model.hierarchy/hierarchy.rule.table \
-lmdir ../work/lm/ \
-nref 4 \
-ngram 5 \
-out ../work/config/NiuTrans.hierarchy.user.config

解码器配置文件截图：

```
#>>>>>>>>>>>>>>>>>>>>>>
#>>> runtime parameters
# number of MERT iterations
param="nround"                          value="12"
# order of n-gram language model
param="ngram"                           value="5"
# maximum phrase length
param="maxphraselength"                 value="7"
# use punctuation pruning (1) or not (0)
param="usepuncpruning"                  value="1"
# use cube-pruning (1) or not (0)
param="usecubepruning"                  value="1"
# use maxent reordering model (1) or not (0)
param="use-me-reorder"                  value="1"
# use msd reordering model (1) or not (0)
param="use-msd-reorder"                 value="1"
# number of threads
param="nthread"                         value="8"
# how many translations are dumped
param="nbest"                           value="30"
# output OOV words
param="outputoov"                       value="0"
# output source words that are explicitly deleted
param="outputnull"                      value="0"
# beam size (or beam width)
param="beamsize"                        value="50"
# beam scale. This parameter controls the number of hypotheses
# that are computed in decoding.
# e.g., beamscale=3 means we search over 3 * beamsize hyphotheses
param="beamscale"                       value="1"
# number of references of dev. set
param="nref"                            value="4"
# allow null-translation (i.e. word-deletion)
param="usenulltrans"                    value="0"
# allow sequence of null-translations
param="snulltrans"                      value="1"
# normalize output (1) or not (0)
param="normalizeoutput"                 value="0"
# distortion limit
param="maxdd"                           value="10"
```

解码器配置文件参数说明：

参数	含义
Ngram-LanguageModel-File	语言模型二进制文件
Target-Vocab-File	语言模型词汇表
SCFG-Rule-Set	层次短语翻译模型文件
Punct-Vocab-File	解码器标点切分文件（可自定义）
nround	MERT 轮数
ngram	语言模型阶数（最好设置为 5）
maxphraselength	解码加载最大源语言端短语长度
usepuncpruning	解码标点切分开关（加速解码方法）
usecubepruning	cube-pruning 开关（加速解码方法）
nthread	线程数
nbest	生成翻译候选个数
outputoov	输出 OOV 开关
outputnull	输出空翻译开关
beamsize	解码 beam 大小
nref	开发集参考答案个数
usenulltrans	使用空翻译开关
maxdd	最大调序距离
weights	权重（每一维特征使用空格切分）
ranges	开发集特征值调整范围

4.7 规则过滤

由于训练生成翻译模型和调序模型比较大，无法全部转载到内存中，因为调优和解码是在开发集上，可以用开发集过滤翻译模型和调序模型。

规则过滤用到的程序：NiuTrans.PhraseExtractor

使用说明：首先进入 NiuTrans_CWMT/目录，然后在命令行下输入：

```
$> bin/NiuTrans.PhraseExtractor --FILPD \
        -dev      开发集和测试集 \
        -in       层次短语规则表 \
        -out      过滤后的层次短语规则表 \
        -maxlen   层次短语规则中规则的最大长度 \
        -rnum     开发集中答案的个数
示例：
$> cat   data/dev/dev.txt   data/test.txt > data/dev/dev_and_test.txt
$> bin/NiuTrans.PhraseExtractor --FILPD \
```

```
        -dev    data/dev/ dev_and_test.txt \
        -in     work/model.hierarchy/hierarchy.rule.table \
        -out    work/ model.hierarchy/hierarchy.rule.table.filterDevAndTest \
        -maxlen 10 \
        -rnum   4
```

注意：过滤完毕后需要将层次短语表的参数改为过滤后的。

$> vim work/config/NiuTrans.hierarchy.user.config

param = “SCFG-Rule-Set”

value=”../work/model.hierarchy/hierarchy.rule.table”

修改为

param=“SCFG-Rule-Set” value=”../work/model.hierarchy/hierarchy.rule.table.filterDevAndTest”

param="nbest"设为 value="30"

param="beamsize"设为 value="50"

本次 baseline 系统中-maxlen 参数设为 10，-rnum 参数设为 4。

4.8 权重调优

NiuTrans.Hierarchy 英汉基线系统直接在处理好的标准开发集上调优，调优后把最佳权重写到配置文件里面，测试集解码时，直接用开发集上的最优权重。

权重调优脚本：NiuTrans-hierarchy-mert-model.pl

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl NiuTrans-hierarchy-mert-model.pl \
        -config 配置文件\
        -dev    处理好的开发集 \
        -nref   n(参考答案的个数) \
        -round  n(mert 的轮数)
        -log    生成调优文件的日志
```

示例：

```
$> perl NiuTrans-hierarchy-mert-model.pl \
        -config ../work/config/NiuTrans.hierarchy.user.config\
        -dev    ../data/dev/dev.txt \
        -nref    4 \
        -round  2 \
        -log    ../work/mert-model.log
```

注：想设置更多权重调优命令，请参见脚本 usage 用法：perl NiuTrans-hierarchy-mert-model.pl 回车。

本次英汉层次短语 baseline 系统最后使用的参数如下：

param="weights" value="1.79125 1.8879 1.2586 0.36345 0.48365 0.3631 -0.80555 0.6 -0.145 -2.23495 1.4811 0 0 0 0 0 0"

4.9 解码

解码这一步在调优之后，直接利用开发集训练好的权重，在测试集上解码。

解码脚本：perl NiuTrans-hierarchy-decoder-model.pl 输入：测试集，配置文件，输出：1best.out

使用说明：首先进入 NiuTrans_CWMT/scripts 目录，然后在命令行下输入：

```
$> perl   NiuTrans-hierarchy-decoder-model.pl \
          -config       配置文件 \
          -test         预处理好的测试集 \
          -opnull       是否输出 oov（默认为 1 输出，0 为不输出） \
          -output       翻译结果
示例：
$> perl   NiuTrans-hierarchy-decoder-model.pl \
          -config       ../work/config/NiuTrans.hierarchy.user.config \
          -test         ../data/ test/test.txt \
          - opnull      0 \
          -output       1best.out
```

4.10 后处理

Baseline 解码出来的结果在评价之前需要先进行后处理。系统最终的输出结果是分词的，而且中文标点符号变为英文的符号，对应关系如下：

系统输出翻译结果	中文标点
. . .	…
.	。
<<	《
>>	》
“”	“ ”

在后处理过程中我们会做以下处理：

1. 首先把系统输出结果中文之间的空格去掉。
2. 按照上面的表格，把系统输出结果的标点转换成对应的中文标点。以上两步我们可以通过系统提供的脚本（NiuTrans_CWMT/tools/generate-xml-tools/punc.postedit.pl）进行处理。

3. 按照往年的评价规范，我们把系统的输出结果进行 A3 区的全角到半角的转换，这个步骤可以用我们提供的“英汉 A3 区全半角转换工具”（NiuTrans_CWMT/tools/en2ch.A3）完成，使用方法见 readme.txt。

另外需要注意的一点是，汉语中的句号、书名号、双引号和顿号为全角，无法替换成半角，所以这部分符号在进行 A3 区全半角转换的时候没有进行改动。

4.11 评价

采用 CWMT2013 官方评价程序，只需将经过后处理的结果变成评价需要的 xml 格式文件，然后输入到官方评价程序即可。我们搭建系统的时候实际上是先进行上一步后处理的前两步，即先把系统输出结果中文之间的空格去掉、标点转换，然后生成 xml 文件。这里可以用我们提供 generate-xml-tools（NiuTrans_CWMT/tools/generate-xml-tools）工具，它可以完成后处理前两步，并生成 xml 文件。generate-xml.pl 是用来生成 xml 的脚本，使用方法如下：

```
$> perl   generate-xml.pl \
          -flist      系统输出的翻译结果文件 \
          -task       英汉新闻翻译默认为“en_zh_news_trans” \
          -1best      \
          -evaluate \
          -rlist       参考答案文件\
          -nref        参考译文数
示例：
$> perl   generate-xml.pl \
          -flist      1best.dev.out \
          -task       en_zh_news_trans \
          -1best      \
          -evaluate \
          -rlist       ref.dev.E2C.txt \
          -nref        4
```

运行之后生成三个 xml 文件：

- src.xml：源语言 xml 文件
- ref.xml：参考答案 xml 文件
- 1best.xml： 系统输出翻译结果 xml 文件

如果出现结果文件与标准答案标点符号不统一的情况，则可用 A3 区全半角转换工具进行转换统一标点符号。

最后得到的 xml 文件就是评价需要的文件格式。mteval_sbp 是 CWMT 官方提

供的评价工具，我们只要把以上三个文件输入到程序中，就可以得到评价结果1best_result，命令如下：

mteval_sbp -c -r ref.xml -s src.xml -t 1best.xml >1best_result

本次英汉层次短语基线系统在 dev 上的 BLEU_SBP5 值：0.3286。

注：英汉层次短语基线翻译系统的标准配置文件见 NiuTrans_1.3.0_CWMT2013\config\CWMT2013-e2c 目录下：

训练层次短语规则配置文件：NiuTrans.phrase.train.model.CWMT.config

解码器配置文件：NiuTrans.phrase.CWMT.config

附件 1

数据名称	原始数据	乱码过滤掉数据	乱码过滤后数据	长度比过滤掉	长度比过滤后行数
Datum	1000004	42251	957753	13593	944160
HIT-IR	100000	433	99567	1096	98471
HIT-MT 目录下	52227	2121	50106	455	49651
HIT-MT-canyin	11352	522	10830	96	10734
HIT-MT-jiaotong	11869	858	11011	68	10943
HIT-MT-lvyou	11408	627	10781	88	10693
HIT-MT-shangwu	9605	46	9559	107	9452
HIT-MT-tiyu	7993	68	7925	96	7829
ICT-web-A	936654	189	936465	14588	921877
ICT-web-B	977176	4138	973038	13230	959808
NEU	1000000	86	999914	26544	973370
XMU-Movie-Subtitle	176148	322	175826	6714	169112
CLDC-LAC-2003-006	200082	640	199442	5154	194288
CLDC-LAC-2003-004 目录下	252329	7822	244507	8598	235909
CLDC-LAC-2003-004-CASIA	105476	5030	100446	5109	95337
CLDC-LAC-2003-004-ICT-art	4006	517	3489	102	3387
CLDC-LAC-2003-004-ICT-eco	4059	129	3930	66	3864
CLDC-LAC-2003-004-ICT-env	4034	39	3995	54	3941
CLDC-LAC-2003-004-ICT-law	7573	51	7522	200	7322
CLDC-LAC-2003-004-ICT-life	62997	365	62632	1073	61559
CLDC-LAC-2003-004-ICT-liter	5373	147	5226	149	5077
CLDC-LAC-2003-004-ICT-news	4159	436	3723	94	3629
CLDC-LAC-2003-004-ICT-oral	27719	58	27661	567	27094
CLDC-LAC-2003-004-ICT-others	4026	15	4011	107	3904
CLDC-LAC-2003-004-ICT-politics	4063	148	3915	232	3683
CLDC-LAC-2003-004-ICT-sci	4266	536	3730	89	3641
CLDC-LAC-2003-004-ICT-sports	4074	231	3843	82	3761
CLDC-LAC-2003-004-ICT-trade	10504	120	10384	674	9710
2003-863-001	606	2	604	2	602