# 语言模型训练流程

## 1、工具安装

### 1.1 srilm依赖库的安装

参考网址：

<http://blog.csdn.net/ownfire/article/details/47276219>

安装需要的文件：SRILM\_depen\libiconv-1.14

首先打开SRILM\_depen\libiconv-1.14\srclib/stdio.in.h ，然后修改第698行为如下（“-”开始行为删除，“+”开始行为增加）：

-\_GL\_WARN\_ON\_USE (gets, "gets is a security hole - use fgets instead");

+#if defined(\_\_GLIBC\_\_) && !defined(\_\_UCLIBC\_\_) && !\_\_GLIBC\_PREREQ(2, 16)

+ \_GL\_WARN\_ON\_USE (gets, "gets is a security hole - use fgets instead");

+#endif

按照INSTALL 文件中的安装说明进行安装：

./Configure && make && make install

### 1.2 Srilm安装

#### 1.2.1 安装过程

Srilm 安装参考文档：

<http://www.cnblogs.com/tongyan/p/3214577.html>

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

2.解压：tar zxvf srilm.tar.gz

3.修改 Makefile 文件（SRILM 目录下）

            找到：

　　　　　# SRILM = /home/speech/stolcke/project/srilm/devel

　　　　　另起一行输入 SRILM 的安装路径  SRILM = $(PWD)

　　　　   找到：

　　　　　MACHINE\_TYPE := $(shell $(SRILM)/sbin/machine-type)

　　　　　在其前加＃注释掉，并另起一行输入：

　　　　　MACHINE\_TYPE := i686-gcc4

4.修改srilm/common/Makefile.machine.i686-gcc4

　　　　    找到：

GCC\_FLAGS = -march=athlon64 -m64 -liconv -Wall -Wno-unused-variable -Wno-uninitialized

　　　　　GCC\_FLAGS = -mtune=pentium3 -Wall -Wno-unused-variable -Wno-uninitialized

　　　　　CC = $(GCC\_PATH)gcc $(GCC\_FLAGS) -Wimplicit-int

　　　　　CXX = $(GCC\_PATH)g++ $(GCC\_FLAGS) -DINSTANTIATE\_TEMPLATES

　　　　　这里是为了告诉 SRILM 系统使用的 compiler(c 和 c++)，符合安装情况，不需要修改。如果是 64 位 CPU，需要做相应调整

　　　       找到：

　　　　　TCL\_INCLUDE =

　　　　　TCL\_LIBRARY =

　　　　　修改为

　　　　　TCL\_INCLUDE =

　　　　　TCL\_LIBRARY =

　　　　　NO\_TCL = X

　　　　   找到：

　　　　　GAWK = /usr/bin/awk

　　　　　修改为

　　　　　GAWK = /usr/bin/gawk

5.安装SRILM

　　　　编译：srilm目录下：make World

#### 1.2.2修改环境变量：

修改 文件： /root/.bash\_profile 然后执行 source 命令加载此配置

PATH=$PATH:$HOME/bin

SRILM=/root/srilm\_test\_szm/srilm

PATH=$PATH:$SRILM/bin:$SRILM/bin/i686-m64

export SRILM

export PATH

#### 1.2.3 注意事项

**glibcxx 库错误**

如果执行 ngram-count 命令的时候，出现如下错误类型：

./bin/i686-m64/ngram-count: /lib64/libstdc++.so.6: version `GLIBCXX\_3.4.20' not found (required by ./bin/i686-m64/ngram-count)

修改 /etc/ld.so.conf 文件内容：

include ld.so.conf.d/\*.conf

/usr/local/lib64

/usr/local/lib

/usr/lib

然后调用 ldconfig 重新加载配置即可。

【 /usr/local/lib64 目录下的 libstdc++.so.6.0.20】

### 1.3 分词工具

分词工具下载地址：

<https://pypi.python.org/pypi/jieba/>

安装方法：在 jieba-0.37 目录下执行 python setup.py install 即可。

* Jieba基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)
* 采用了动态规划查找最大概率路径, 找出基于词频的最大切分组合
* 对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法

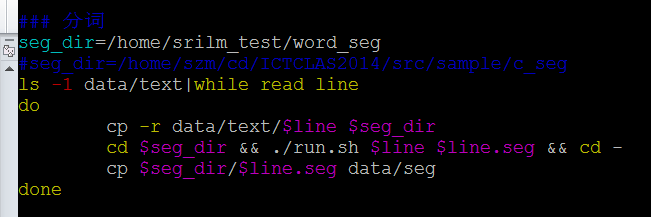
## 2、训练流程

### 2.1使用方法

lm\_train 为训练环境根目录，首先将训练语料文本放到 data/text目录下（可多个文件），然后运行 nohup ./run.sh & 即可，生成的语言模型在data/lm目录下：out\_lm\_3.arpa（三元模型）、out\_lm\_3\_prune.arpa（三元模型剪枝后）、out\_lm\_4.arpa（四元模型）。

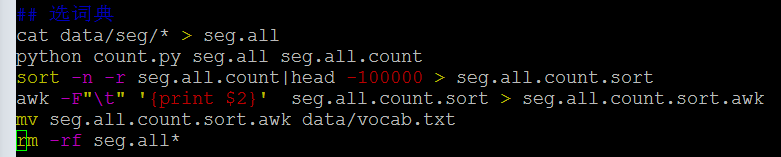
其中kaldi解码器，画图时需要使用out\_lm\_3\_prune.arpa模型，解码时需要使用out\_lm\_4.arpa大模型。

### 2.2分词工具



此段脚本，是将data/text中的每个文件拷贝到word\_seg目录下，然后由jieba分词工具进行分词，结果保存在data/seg文件中。如果提供的原始文件不需要分词，那么此过程不需要执行，将分词的文本放到data/seg中即可。

### 2.3 词典选取



此段脚本，调用count.py工具，对分词后的文本进行词频统计，然后从中选出指定个数的高频词来作为词典vocab.txt。

### 2.4模型训练

ngram-count -order 3 -kndiscount -interpolate -unk -map-unk "<UNK>" -limit-vocab -vocab $vocab -text $full\_corpus -lm $trigram\_lm || exit 1

此段脚本为：使用给定的文本、词典，来训练语言模型的脚本。其中，

-order 3 :指定为3元模型训练。

-kndiscount –interpolate :设定平滑算法。

-unk -map-unk "<UNK>" :未知词标记为“<UNK>”

-limit-vocab : 未知词全部按照UNK来统计词频。

-vocab $vocab ：指定词典

-text $full\_corpus ：指定用于训练的文本

-lm $trigram\_lm ：指定输出的语言模型

### 2.5 语言模型裁剪

ngram -prune $prune\_thresh\_small -lm $trigram\_lm -write-lm $trigram\_pruned\_small || exit 1

如果训练出的语言模型占用空间比较大，可以使用此命令进行裁剪成小模型。其中，

-prune $prune\_thresh\_small ：设定裁剪的阈值

-lm $trigram\_lm ：输入的需要裁剪模型

-write-lm $trigram\_pruned\_small ：输出裁剪后的模型