

第三讲 词对齐

学习目标：学会使用词对齐工具 GIZA++并自行开发词对齐对称化程序。

词对齐是统计机器翻译系统构建的第二步，经过第一步双语数据预处理后，得到分词后的双语数据。而词对齐的任务就是要得到中英文词语的对应关系。

本讲学习内容：

● GIZA++的使用

GIZA++是 GIZA (SMT 工具包 EGYPT 的一个组成部分) 的扩展，扩展部分主要由 Franz Josef Och 开发。GIZA++主要算法是 IBM model、HMM 等。

1、GIZA++运行环境：Linux，并预装软件 gcc、g++。

2、GIZA++下载地址：

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

3、编译：

a)首先进入到 GIZA++根目录

b)解压包，指令：tar zxvf giza-pp-v1.0.7.tar.gz

c)进入到解压后的目录，指令：cd giza-pp

d)编译，指令：make

4、GIZA++运行：

a)新建目录 Alignment，并将编译后的 GIZA++-v2 目录下的“GIZA++”、“snt2cooc.out”、“plan2snt.out”文件和 mkcls-v2 目录下的“mkcls”文件，拷贝到 Alignment 目录下，同时将预处理后的文件 c.txt 和 e.txt 作为 GIZA++工具的输入文件放到其中。

b)在终端依次运行以下命令，获得下一步需要的文件：c2e.A3.final 和 e2c.A3.final。

```
$> ./plain2snt.out e.txt c.txt
$> ./snt2cooc.out c.vcb e.vcb c_e.snt> cooc.ce
$> ./snt2cooc.out e.vcb c.vcb e_c.snt > cooc.ec
$> ./mkcls -m2 -c80 -n10 -pe.txt -Ve.vcb.classes opt
$> ./mkcls -m2 -c80 -n10 -pc.txt -Vc.vcb.classes opt
$> ./GIZA++ -S c.vcb -T e.vcb -C c_e.snt -CooccurrenceFile cooc.ce -O c2e
$> ./GIZA++ -S e.vcb -T c.vcb -C e_c.snt -CooccurrenceFile cooc.ec -O e2c
```

● 词对齐对称化

由于 GIZA++程序中，原语=>目标语和目标语=>原语的对齐过程是彼此独立的，因此会产生两个对齐文件，词对齐对称化的任务就是通过一定的算法合并这两个对齐文件。

1、输入：GIZA++运行目录中的 c2e.A3.final 文件和 e2c.A3.final 文件，格式如下：

c2e.A3.final 文件的一个句对：

```
# Sentence pair (1) source length 5 target length 7 alignment score : 3.53407e-09
the weather is very good today .
NULL ({ 1 3 6 }) 今天 ({ }) 天气 ({ 2 }) 真 ({ 4 }) 好 ({ 5 }) . ({ 7 })
```

e2c.A3.final 文件的一个句对：

```
# Sentence pair (1) source length 7 target length 5 alignment score : 5.24219e-08
今天 天气 真 好 .
NULL ({ }) the ({ 2 }) weather ({ 2 }) is ({ }) very ({ 3 }) good ({ 4 }) today ({ 1 }) . ({ 5 })
```

其中，第三行大括号里面的数字，代表该大括号前面的词语对应的目标语的词的位置。

如：“天气 ({ 2 })”中的“2”表示对应的英语中的“weather”。

“weather ({ 2 })”中的“2”表示对应的汉语中的“天气”。

因此，“天气”和“weather”构成了一个双向对齐“天气” \leftrightarrow “weather”。

“today ({ 1 })”中的“1”表示对应的汉语中的“今天”。

而“今天”后的({})为空，并未与“today”对应。

因此，“today”和“今天”就构成了一个单向对齐“today” \Rightarrow “今天”。

另外，“NULL ({ 1 3 6 })”表示英文中“the”“is”“today”对空，即形成空对齐“the” \Rightarrow “NULL”，“is” \Rightarrow “NULL”，“today” \Rightarrow “NULL”。

2、输出：将两个输入文件通过算法合并为对齐文件，文件中每一行格式如下：

```
0-5 1-0 1-1 2-3 3-4 4-6
```

其中，0-5 表示经过算法合并后，中文中第 0 个词“今天”与英文中第 5 个词“today”对齐。

3、词对齐对称化算法：

算法流程如下：

Matrix[src_len][trg_len];

neighboring = (上,下,左,右,左上,左下,右上,右下);

寻找对齐节点，将双向对齐节点加入 Alignment

循环直到没有新节点出现

遍历 Alignment，查看该节点的 neighboring

若某一 neighboring 单向对齐，且该点的原语或目标语未双向对齐
则加入该节点到 Alignment

遍历 Matrix

如果存在某节点单向对齐，且该点的原语或目标语未双向对齐
则加入该节点到 Alignment

该算法与 Philipp Koehn 的 GROW-DIAG-FINAL 算法类似。

4、算法说明：

以上面的句对作为示例：

中文到英文：

Sentence pair (1) source length 5 target length 7 alignment score : 3.53407e-09

the weather is very good today .

NULL ({ 1 3 6 }) 今天 ({ }) 天气 ({ 2 }) 真 ({ 4 }) 好 ({ 5 }) . ({ 7 })

英文到中文：

Sentence pair (1) source length 7 target length 5 alignment score : 5.24219e-08

今天 天气 真好 .

NULL ({ }) the ({ 2 }) weather ({ 2 }) is ({ }) very ({ 3 }) good ({ 4 }) today ({ 1 }) . ({ 5 })

- 1) 首先，找到对齐的词语对，将所有双向对齐加入 **Alignment**。如图（矩阵中实心方格表示双向对齐，虚线空心方格表示单向对齐）：

	the	weather	is	very	good	today	.
今天							
天气							
真							
好							
.							

- 2) 遍历 **Alignment**，对其邻居进行检测，如果邻居中存在单向对齐的点，并且该点的两个词中存在一个没有与任何词双向对齐，则将该点加入 **Alignment**。循环这个操作直到没有新节点加入。如上图，当遍历到“weather-天气”时，发现该点左边存在单向对齐点“the-天气”，且“the”尚未与任何词双向对齐，则将该点加入 **Alignment**。

	the	weather	is	very	good	today	.
今天							
天气							
真							
好							
.							

- 3) 最后遍历矩阵中所有节点，如果存在单向对齐的点，并且该点的两个词中**存在**一个没有与任何词双向对齐，则将该点也加入 **Alignment**。如图，当遍历到“today-今天”时，发现该点是一个单向对齐点，并且 “today” 尚未与任何词双向对齐，则将该点加入 **Alignment**。

	the	weather	is	very	good	today	.
今天							
天气							
真							
好							
.							

- 4) 最后获得最终的对齐信息输出到文件，如上图所示，得到的对称化的词对齐信息为：

0-5 1-0 1-1 2-3 3-4 4-6