

自然语言处理作业

第二次大作业 2021. 11. 1

07111908 班 1120193583 张昊杰

电话 18991820738 邮箱 zhj727534681@163.com

一、实验内容：

选择实验 4 进行，即汉英词语自动对齐系统

二、实现平台

本地平台：电脑一：MacBook Air M1，语言选择：Python 和 Cpp，

电脑二：AMD 3800X 16 线程 运行 Centos 便于本地调试。

云端服务器：腾讯云 1 核云服务器，系统为 CentOS，Web 端选择 Django，语言选择为 Python 和 Cpp。

三、实验人员和分工情况

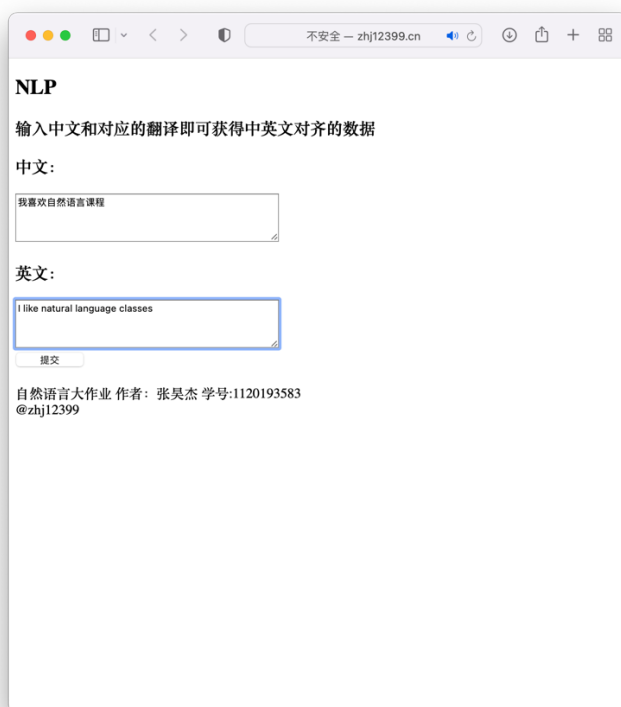
由本人独自完成本项目

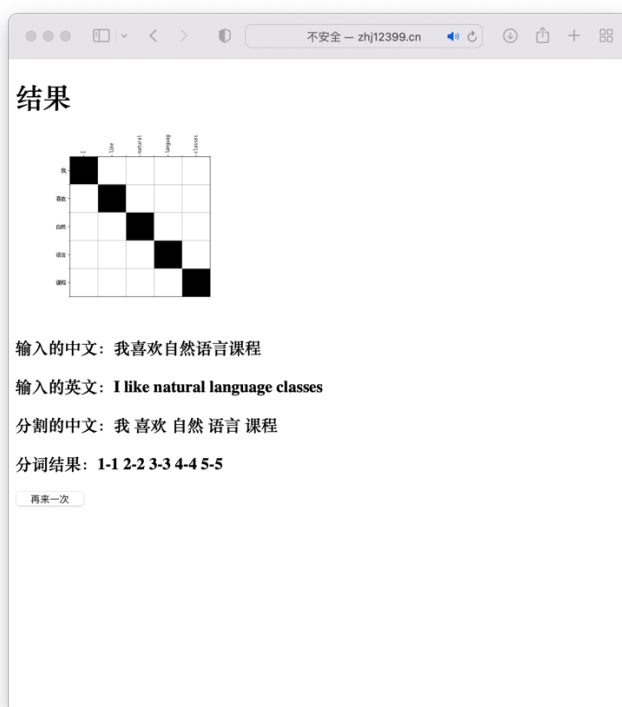
四、项目成果展示

为了便于展示成果，也是满足实验要求，将程序放置在云服务器中建立了一个 Web 应用这样也可以让所有人来访问。

网址为 <http://NLP.zhj12399.cn>，如 DNS 不正确可访问 <http://zhj12399.cn:9200>

同时也录制了一段自己运行的展示视频在文件夹中。





五、核心算法和模块

核心理想：参照了在分词算法中较为流行的 GIZA++ 的远源码作为蒙版，进行进一步的改写与移植来实现我们的平台。Web 端和服务器搭建等不是本课程内容就不介绍了。

下面简要介绍 GIZA++ 算法：

GIZA++完整的实现了（Brown）描述的独立与词集的 IBM-4 对齐模型；实现了 IBM-5.实现了 HMM 对齐模型，并且改进了 IBM-1，IBM-2，和 HMM 模型的概率计算算法。

GIZA++属于开源代码：已在 Github 中开源：<https://github.com/moses-smt/giza-pp>, codechina 中也有加速镜像的项目：<https://codechina.csdn.net/mirrors/moses-smt/giza-pp.git>

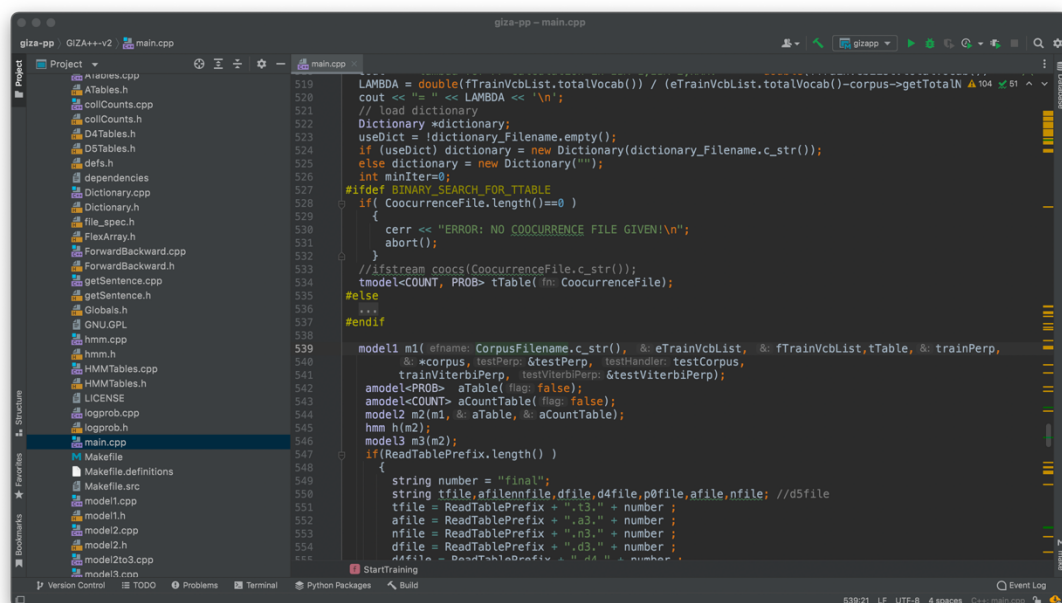
GIZA++主要包含了一下几个程序：

1. GIZA++本身
2. Plain2snt.out 将普通文本转换成 GIZA 格式
3. Snt2plain.out 将 GIZA 格式文本转换成普通文本
4. Snt2cooc.out 用 GIZA 格式文本生成共线文件

GIZA 训练流程：

先从 IBM-1 模型开始，训练结果用来初始化 IBM-2 模型的训练。IBM-2 转换成 IBM-3，然后进行 IBM-3viterbi 训练。

我们下载源码之后阅读当中的源码也可以从源码中看到。从 model1、modal2、HMM、modal3 进行的。



GIZA 的主要使用流程是：

首先 git 下来项目，cd 到 giza-pp 目录下进行 make 编译。

编译完后生成 plain2snt.out、snt2cooc.out、GIZA++、mkcls 四个可执行文件。

首先将准备好的训练的双语语料库分别存放到文件中，

zh.txt:

海洋 是一个 非常 复杂 的事物 。

人类 的健康 也 是一件 非常 复杂 的事情 。

将两者统一起来看起来是一件艰巨的任务。但我想要试图去说明的是即使是如此复杂的情况，也存在一些我认为简单的话题，一些如果我们能理解，就很容易向前发展的话题。这些简单的话题确实不是有关那复杂的科学有了怎样的发展，而是一些我们都恰好知道的事情。接下来我就来说一个。如果老妈不高兴了，大家都别想开心。

en.txt:

```
It can be a very complicated thing , the ocean .
And it can be a very complicated thing , what human health is .
And bringing those two together might seem a very daunting task , but what I 'm
going to try to say is that even in that complexity , there 's some simple
themes that I think , if we understand , we can really move forward .
And those simple themes aren 't really themes about the complex science of what
's going on , but things that we all pretty well know .
And I 'm going to start with this one : If momma ain 't happy , ain 't nobody
happy .
```

然后运行命令进行词对齐。

```
./plain2snt.out zh.txt en.txt
```

得到 en.vcb、zh.vcb、en_zh.snt、zh_en.snt 四个文件

en.vcb / zh.vcb: 字典文件:

```
2 海洋 1
3 是 6
4 一个 2
5 非常 2
6 复杂 4
7 的 12
8 事物 1
9 。 7
10 人类 1
...
```

en_zh.snt / zh_en.snt: 编号表示句对，第一行表示句对出现次数

```
1
2 3 4 5 6 7 8 9
2 3 4 5 6 7 8 9 10 11 12 13
1
10 7 11 12 3 13 14 5 6 7 15 9
14 15 4 5 6 7 8 9 10 16 17 18 19 13
...
```

接下来执行命令生成共现文件:

```
./snt2cooc.out zh.vcb en.vcb zh_en.snt > zh_en.cooc
```

```
./snt2cooc.out en.vcb zh.vcb en_zh.snt > en_zh.cooc
```

```
0 33
0 34
0 35
0 36
0 37
0 38
0 39
0 40
...
```

接下来执行命令生成词类:

```
./mkcls -pzh.txt -Vzh.vcb.classes opt
./mkcls -pen.txt -Ven.vcb.classes opt
```

```
***** 1 runs. (algorithm:TA)*****
```

```
;KategProblem:cats: 100 words: 68
```

```
start-costs: MEAN: 262.907 (262.907-262.907) SIGMA:0
end-costs: MEAN: 190.591 (190.591-190.591) SIGMA:0
start-pp: MEAN: 3.52623 (3.52623-3.52623) SIGMA:0
end-pp: MEAN: 1.95873 (1.95873-1.95873) SIGMA:0
iterations: MEAN: 50117 (50117-50117) SIGMA:0
time: MEAN: 1.468 (1.468-1.468) SIGMA:0
```

en.vcb.classes / zh.vcb.classes: 单词所属类别编号

```
,      26
.      28
:      64
And    29
I      13
If     52
It     49
a      34
about  22
```

en.vcb.classes.cats / zh.vcb.classes.cats: 类别所拥有的一组单词

```
0:$,
1:
2:science,
3:seem,
4:things,
5:some,
6:start,
```

7:task,

执行 **GIZA++**，先在当前目录新建两个输出文件夹 **z2e**、**e2z**，我们会将文件输出到这两个文件夹中。

```
./GIZA++ -S zh.vcb -T en.vcb -C zh_en.snt -CooccurrenceFile zh_en.cooc -o z2e -
OutputPath z2e
./GIZA++ -S en.vcb -T zh.vcb -C en_zh.snt -CooccurrenceFile en_zh.cooc -o e2z -
OutputPath e2z
```

输出的文件：
z2e.perp 困惑度：

#trnsz	tstsz	iter	model	trn-pp	test-pp	trn-vit-pp	tst-vit-pp
5	0	0	Model1	80.5872	N/A	2250.77	N/A
5	0	1	Model1	36.0705	N/A	648.066	N/A
5	0	2	Model1	34.0664	N/A	523.575	N/A
5	0	3	Model1	32.628	N/A	423.928	N/A
5	0	4	Model1	31.5709	N/A	359.343	N/A
5	0	5	HMM	30.7896	N/A	314.58	N/A
5	0	6	HMM	31.1412	N/A	172.128	N/A
5	0	7	HMM	26.1343	N/A	111.444	N/A
5	0	8	HMM	22.177	N/A	79.3055	N/A
5	0	9	HMM	19.0506	N/A	58.4415	N/A
5	0	10	THTo3	32.6538	N/A	37.8575	N/A
5	0	11	Model3	11.1194	N/A	11.944	N/A
5	0	12	Model3	8.93033	N/A	9.50349	N/A
5	0	13	Model3	7.68766	N/A	8.19622	N/A
5	0	14	Model3	6.64154	N/A	7.04977	N/A
5	0	15	T3To4	6.17993	N/A	6.55567	N/A
5	0	16	Model4	6.16858	N/A	6.4715	N/A
5	0	17	Model4	6.0819	N/A	6.39317	N/A
5	0	18	Model4	6.04302	N/A	6.34387	N/A
5	0	19	Model4	5.95066	N/A	6.2234	N/A

z2e.a3.final: i j l m p(i/j, l, m): i 代表源语言 Token 位置；j 代表目标语言 Token 位置；l 代表源语言句子长度；m 代表目标语言句子长度

1 1 8 100 1
5 2 8 100 1
4 3 8 100 1
2 4 8 100 1
5 5 8 100 1
4 6 8 100 1
4 7 8 100 1
0 8 8 100 1

...

z2e.d3.final: 类似于 **z2e.a3.final** 文件，只是交换了 i 和 j 的位置。

2 0 100 8 0.0491948

6 0 100 8 0.950805

3 1 100 8 1

5 2 100 8 1

4 3 100 8 1

2 4 100 8 0.175424

5 4 100 8 0.824576

2 5 100 8 1

4 6 100 8 1

4 7 100 8 1

...

z2e.n3.final: source_id p0 p1 p2 ... pn; 源语言 Token 的 Fertility 分别为 0,1,...,n 时的概率表，比如 p0 是 Fertility 为 0 时的概率。

2 1.22234e-05 0.781188 0.218799 0 0 0 0 0 0

3 0.723068 0.223864 0 0.053068 0 0 0 0 0 0

4 0.349668 0.439519 0.0423205 0.168493 0 0 0 0 0 0

5 0.457435 0.447043 0.0955223 0 0 0 0 0 0 0

6 0.214326 0.737912 0 0.0477612 0 0 0 0 0 0

7 1 0 0 0 0 0 0 0 0 0

8 1.48673e-05 0.784501 0.215484 0 0 0 0 0 0 0

...

z2e.t3.final: s_id t_id p(t_id/s_id); IBM Model 3 训练后的翻译表；p(t_id/s_id)表示源语言 Token 翻译为目标语言 Token 的概率

0 3 0.196945

0 7 0.74039

0 33 0.0626657

2 4 1

3 6 1

4 5 1

5 3 0.822024

5 6 0.177976

6 3 0.593075

...

z2e.A3.final 单向对齐文件，数字代表 Token 所在句子位置（1 为起点）

Sentence pair (1) source length 8 target length 11 alignment score : 8.99868e-08

It can be a very complicated thing , the ocean .

NULL ({ 8 }) 海洋 ({ 1 }) 是 ({ 4 }) 一个 ({ 9 }) 非常 ({ 3 6 7 }) 复杂 ({ 2 5 }) 的 ({ }) 事物 ({ 10 }) 。 ({ 11 })

```
# Sentence pair (2) source length 12 target length 14 alignment score : 9.55938e-12
And it can be a very complicated thing , what human health is .
NULL ({ 9 }) 人类 ({ 2 11 }) 的 ({ }) 健康 ({ 12 }) 也 ({ }) 是 ({ 5 }) 一 ({ }) 件 ({ 13 }) 非常 ({ 4
7 8 }) 复杂 ({ 3 6 }) 的 ({ }) 事情 ({ 1 10 }) 。 ({ 14 })
...
```

z2e.d4.final: IBM Model 4 翻译表

```
# Translation tables for Model 4 .
```

```
# Table for head of cept.
```

```
F: 20 E: 26
```

```
SUM: 0.125337
```

```
9 0.125337
```

```
F: 20 E: 15
```

```
SUM: 0.0387214
```

```
-2 0.0387214
```

```
F: 20 E: 24
```

```
SUM: 0.0387214
```

```
21 0.0387214
```

```
...
```

z2e.D4.final: IBM Model 4 的 Distortion 表

```
26 20 9 1
```

```
15 20 -2 1
```

```
24 20 21 1
```

```
2 20 -2 1
```

```
40 20 -4 1
```

```
22 20 -3 0.0841064
```

```
22 20 9 0.915894
```

```
32 20 28 1
```

```
21 20 24 1
```

```
29 2 -3 0.472234
```

```
29 2 1 0.527766
```

```
5 2 1 0.475592
```

```
...
```

其中的单向对齐文件就是我们所需要的结果，很显然并不直观，并且 **GIZA++** 程序只能运行 **Linux** 中并且全部需要命令行操作非常的繁琐，而且还需要用户自己准备语料库，使用起来也不是很轻松。

以上便是 **GIZA++** 的源码解读和正确使用方法，接下来便是我进行修改和植入迁移的过程。

六、移植过程

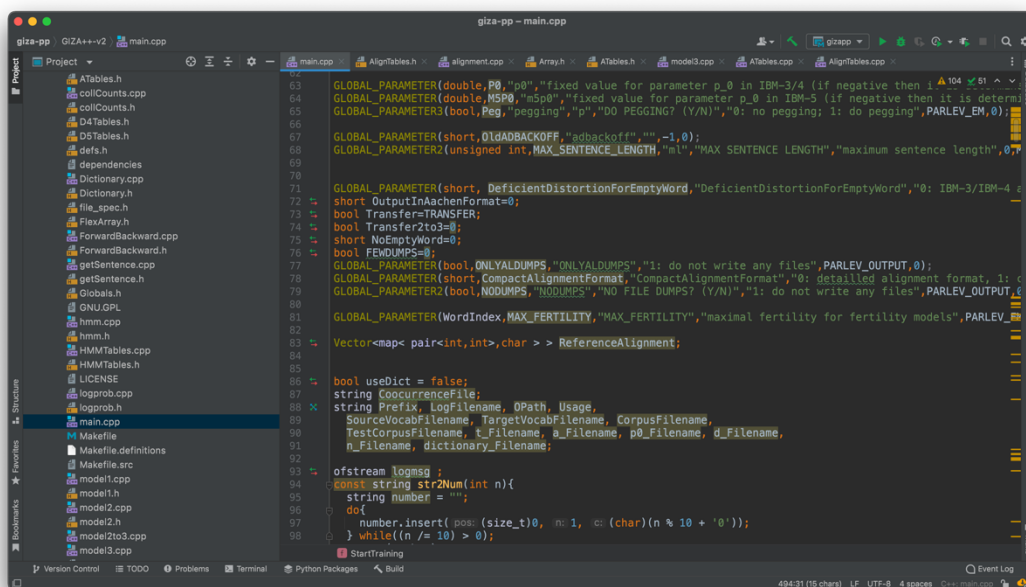
（一）、准备语料库

从开放数据平台 LDC-宾夕法尼亚大学官网下载了一份中英文平行语料库便于我们训练。LDC2019T13 数据集，这个数据集大多为日常口语交流和一些非正式的场合使用的语句，有中文和英文平行的译本大约 10 万句。这样刚好可以拿来训练。下载之后进行整合，去除掉杂余的标点符号和空行等东西之后整理成一个文件，此处的代码详见附录 1。

（二）、修改 GIZA++源码

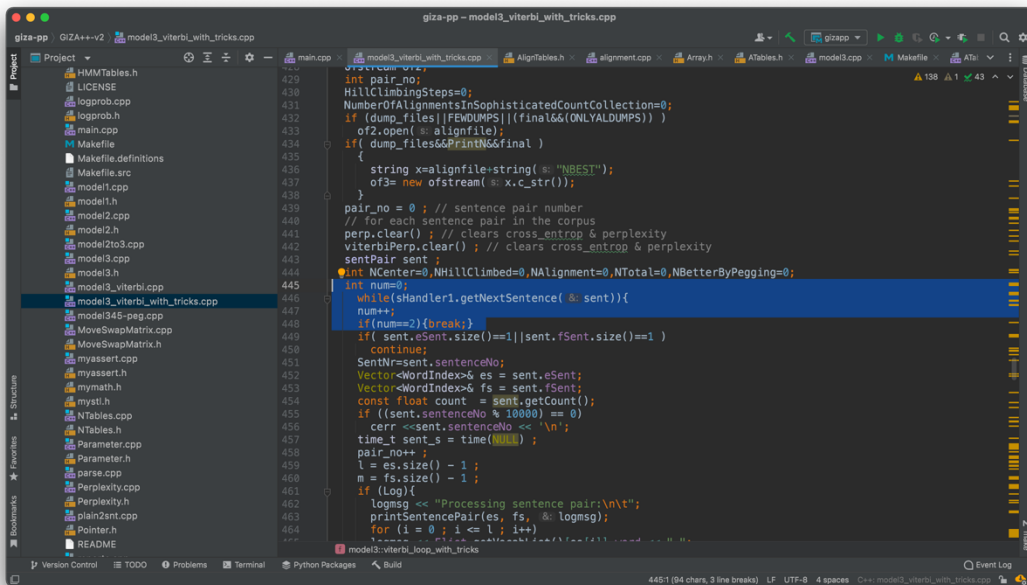
在前几次运行时，我发现这个源码在 16 核处理器中运行的相当缓慢，究其原因有 1.其在运行前会对过长的句子进行裁切成多个短句，并且裁切的过程异常缓慢 2.其在生成预测结果的时候会对所有训练集都生成结果而我们仅仅想要使用训练集那一个预测结果即可

1.在阅读了 sentenceHandler 代码后发现其做的裁剪工作非常复杂不便修改，但是由于 giza 代码是一个 20 年前的代码，当时是处于对当时计算机算力和内存的考量下才使用了这种裁剪后来计算小部分的策略，在当下已经显得没有必要甚至产生负面影响。于是我们放开裁剪的标准，从修改大于 10 个字的句子改为允许 100 个单词以下都不修改。这样大大提升速率。此处第 81 行



```
giza-pp - GIZA++-v2 main.cpp
Project
  ATables.h
  collCounts.cpp
  collCounts.h
  D4Tables.h
  D5Tables.h
  defs.h
  dependencies
  Dictionary.cpp
  Dictionary.h
  file_spec.h
  FlexArray.h
  ForwardBackward.cpp
  ForwardBackward.h
  getSentence.cpp
  getSentence.h
  Globals.h
  GNU.GPL
  hmm.cpp
  hmm.h
  HMMTables.cpp
  HMMTables.h
  LICENSE
  logprob.cpp
  logprob.h
  main.cpp
  Makefile
  Makefile.definitions
  Makefile.src
  model1.cpp
  model1.h
  model2.cpp
  model2.h
  model2to3.cpp
  model3.cpp
main.cpp
  GLOBAL_PARAMETER(double, P0, "p0", "fixed value for parameter p_0 in IBM-3/4 (if negative then it is determined by the training data)", "0: no pegging; 1: do pegging", PARLEV_EN, 0);
  GLOBAL_PARAMETER(double, MSP0, "msp0", "fixed value for parameter p_0 in IBM-5 (if negative then it is determined by the training data)", "0: no pegging; 1: do pegging", PARLEV_EN, 0);
  GLOBAL_PARAMETER3(bool, Peg, "pegging", "p", "DO PEGGING? (Y/N)", "0: no pegging; 1: do pegging", PARLEV_EN, 0);
  GLOBAL_PARAMETER(short, OldADBACKOFF, "adbackoff", "", "-1, 0");
  GLOBAL_PARAMETER2(unsigned int, MAX_SENTENCE_LENGTH, "nl", "MAX SENTENCE LENGTH", "maximum sentence length", 0);
  GLOBAL_PARAMETER(short, DeficientDistortionForEmptyWord, "DeficientDistortionForEmptyWord", "0: IBM-3/IBM-4 a
  short OutputInAachenFormat=0;
  bool Transfer=TRANSFER;
  bool Transfer2to3=0;
  short NoEmptyWord=0;
  bool FENDUMPS=0;
  GLOBAL_PARAMETER(bool, ONLYVALDUMPS, "ONLYVALDUMPS", "1: do not write any files", PARLEV_OUTPUT, 0);
  GLOBAL_PARAMETER(short, CompactAlignmentFormat, "CompactAlignmentFormat", "0: detailed alignment format, 1: c
  GLOBAL_PARAMETER2(bool, NODUMPS, "NODUMPS", "NO FILE DUMPS? (Y/N)", "1: do not write any files", PARLEV_OUTPUT, 0);
  GLOBAL_PARAMETER(WordIndex, MAX_FERTILITY, "MAX_FERTILITY", "maximal fertility for fertility models", PARLEV_EN
  Vector<map< pair<int,int>, char > > ReferenceAlignment;
  bool useDict = false;
  string CoccurrenceFile;
  string Prefix, LogFilename, OPath, Usage,
  SourceVocabFilename, TargetVocabFilename, CorpusFilename,
  TestCorpusFilename, t_Filename, o_Filename, p0_Filename, d_Filename,
  n_Filename, dictionary_Filename;
  ofstream logmsg;
  81: 80: string str2Num(int n){
  string number = "";
  do{
    number.insert(position(size_t)0, (n / 10), (char)(n % 10 + '0'));
  } while((n /= 10) > 0);
  StartTraining
494:31 (15 chars) LF UTF-8 4 spaces C++ main.cpp
```

2.在 modal3 最后生成的时候会生成所有训练集的对齐结果，我们只允许他进行我们的预测句子的生成。



(三) 可视化展示

由于训练结果是一个单项对齐，且为一堆文字信息不便于清晰直观的展示。所以我们需要进行对称化和可视化展示。

这里使用卡内基梅隆大学介绍的 **grow-diag-final-and** 方法进行对称化然后使用 **matplotlib** 进行绘图，这里代码见附录 2，3

首先是双向对齐：

1-1 2-4 3-1 3-9 4-3 4-6 4-7 5-2 5-5 7-10 8-11

1-2 1-11 1-12 3-2 4-13 5-5 6-13 7-13 8-4 8-7 8-8 9-3 9-6 9-10 11-1 12-14

1-2 2-2 2-25 3-2 3-26 4-2 4-11 5-36 6-6 6-29 7-8 8-22 9-22 10-2 12-7 12-21 12-42 12-46 13-1 14-23 15-15

15-19 16-16 16-20 17-25 18-29 19-24 19-31 20-6 23-5 23-30 24-8 25-5 25-10 26-9 26-14 28-4 29-3 30-9 31-

5 31-43 32-3 33-35 34-36 34-45 35-33 37-13 37-44 38-17 39-3 40-16 40-18 41-30 41-34 42-41 43-5 44-17

45-15 45-16 45-17 46-24 47-29 47-38 48-27 48-39 48-40 49-32 51-31 52-47

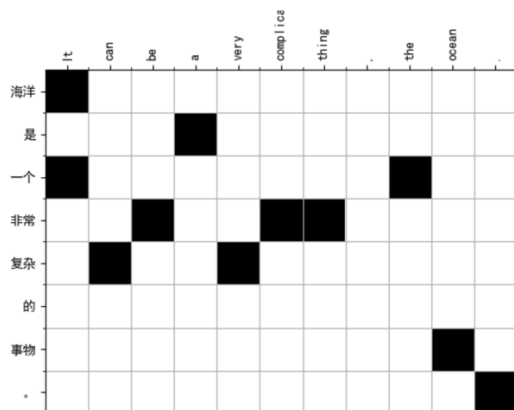
1-5 1-23 1-25 2-4 2-8 4-7 4-19 4-22 5-9 6-6 8-5 9-26 10-14 12-9 13-13 14-6 14-20 15-5 15-17 16-15 17-3

18-2 18-16 19-10 20-2 21-15 21-21 22-6 23-10 24-11 24-12 24-13 24-24 26-1 27-27

1-15 2-2 3-2 3-3 3-4 3-5 4-12 5-7 6-1 6-21 7-20 8-19 8-20 9-8 10-14 10-18 11-7 11-9 11-10 11-11 12-14

13-4 14-13 15-14 16-6 16-13 17-3 18-6 18-17 19-21

然后绘图：

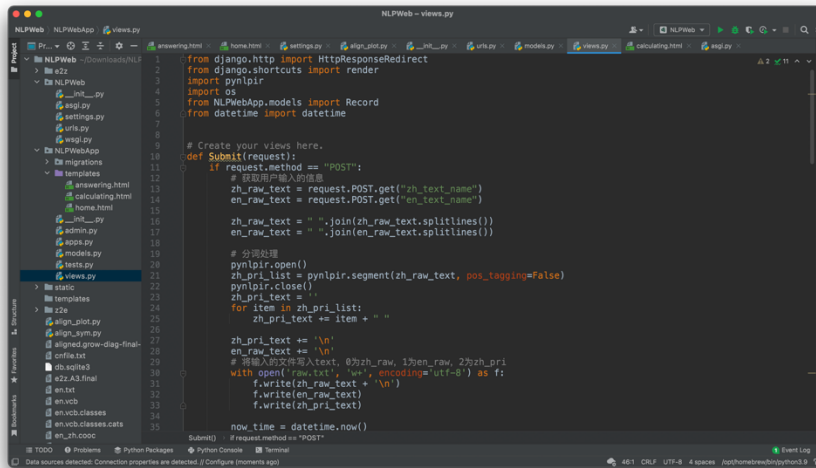


这样我们一套优化下来就可以得到一个较为优秀的结果。

（四）放置在 Web 中便于所有人来浏览

由于前面这几步代码都使用了 Python 来运行，所以我选择了使用 Django 框架作为 Web 服务器，将这些代码都能成功的穿起来并跑在服务器上，这部分与词课程无关我就不介绍了，代码在文件夹中，由于各个机器配置的环境不同，可能无法正常运行。如想体验 Web 程序可至我的个人网站去体验：

NLP.zhj12399.cn



```
from django.http import HttpResponseRedirect
from django.shortcuts import render
import pynlpir
import os
from NLPWebApp.models import Record
from datetime import datetime

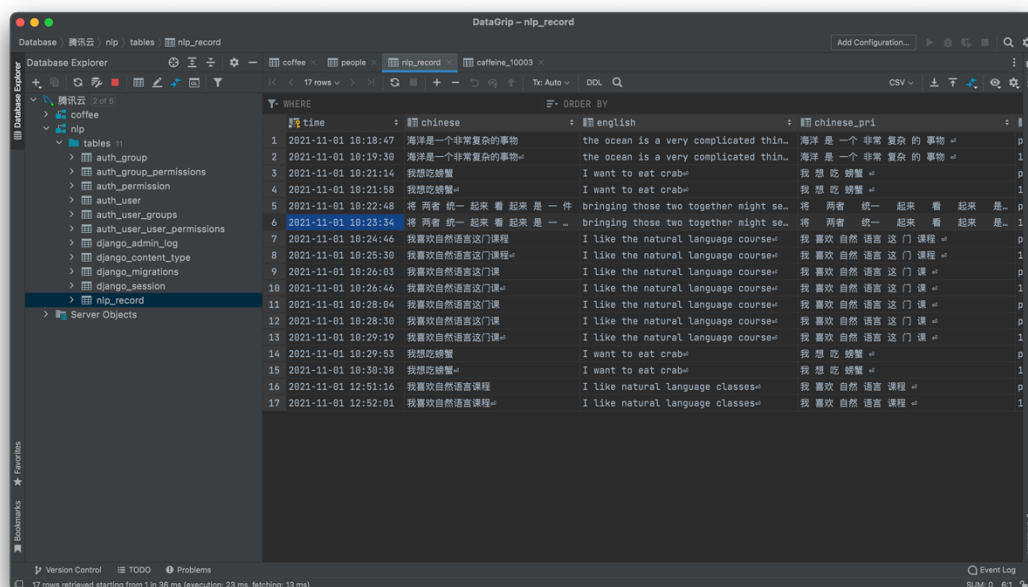
# Create your views here.
def Submit(request):
    if request.method == "POST":
        # 获取用户输入的信息
        zh_raw_text = request.POST.get("zh_text_name")
        en_raw_text = request.POST.get("en_text_name")
        zh_raw_text = "\n".join(zh_raw_text.splitlines())
        en_raw_text = "\n".join(en_raw_text.splitlines())

        # 分词处理
        pynlpir.open()
        zh_pri_list = pynlpir.segment(zh_raw_text, pos_tagging=False)
        pynlpir.close()
        zh_pri_text = ""
        for item in zh_pri_list:
            zh_pri_text += item + " "

        zh_pri_text = "\n"
        en_raw_text += "\n"
        # 将输入的文件与文本，@为zh_raw, 15en_raw, 2为zh_pri
        with open("nlp_record.txt", "w+", encoding="utf-8") as f:
            f.write(zh_raw_text + "\n")
            f.write(en_raw_text + "\n")
            f.write(zh_pri_text)

        now_time = datetime.now()
        Submit() # 记录到数据库
```

同时网站中添加了记录每位用户输入和输出的环节，可以记录用户的输入输出在后期可以将这些用户的记录作为原始训练集加入到我的训练集中来扩充我的语料库



time	chinese	english	chinese_pri
2021-11-01 10:18:47	海洋是一个非常复杂的事物	the ocean is a very complicated thin...	海洋 是一个 非常 复杂 的 事物
2021-11-01 10:19:30	海洋是一个非常复杂的事物	the ocean is a very complicated thin...	海洋 是一个 非常 复杂 的 事物
2021-11-01 10:21:14	我想吃螃蟹	I want to eat crab	我 想 吃 螃 蟹
2021-11-01 10:21:58	我想吃螃蟹	I want to eat crab	我 想 吃 螃 蟹
2021-11-01 10:22:48	将两者统一起来看起来是一件	bringing those two together might se...	将 两 者 统 一 起 来 看 起 来 是 一
2021-11-01 10:23:34	将两者统一起来看起来是一件	bringing those two together might se...	将 两 者 统 一 起 来 看 起 来 是 一
2021-11-01 10:24:46	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:25:30	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:26:03	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:26:46	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:28:04	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:28:39	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:29:19	我喜欢自然语言这门课	I like the natural language course	我 喜 欢 自 然 语 言 这 门 课
2021-11-01 10:29:53	我想吃螃蟹	I want to eat crab	我 想 吃 螃 蟹
2021-11-01 10:30:38	我想吃螃蟹	I want to eat crab	我 想 吃 螃 蟹
2021-11-01 10:51:16	我喜欢自然语言课程	I like natural language classes	我 喜 欢 自 然 语 言 课 程
2021-11-01 12:52:01	我喜欢自然语言课程	I like natural language classes	我 喜 欢 自 然 语 言 课 程

（五）增添分词程序

由于考虑到用户输入的中文不会是分词的结果，所以我们需要程序来进行分词的操作，这里我是用中科大的 pynlpir 程序作为 python 程序中的内嵌程序，将用户输入的完整的中文句子分隔开，也使得用户使用起来很方便。

七、总结展望反思

1. 第一次很完整的算是阅读了很厉害的算法的源码也是感受到了前辈的心血，很严谨的代码风格和踏实的作风，收获还是不小。也是第一次尝试动手修改源代码并且手动去编译他们，感觉非常新鲜也是对自己能做成这件事感到骄傲。
2. 还有一处我认为目前应该对速度影响较小的优化地方，还没有进行优化，有待未来去完善，最开始生成词典的那里，其实可以在词典里直接去查看是否有当前输入的句子的那些词，而不是重新将所有句子都输入到程序中去生成词典，目前来看在单核处理器上生成词典的时间几乎是瞬时完成但这也算是一个可以优化的点。
3. 很精明的加入了对用户输入的信息去记录并且计算生成结果，这样也便于扩充训练集。
4. 对结果只有一个可视化的展示，并没有一个评判标准来评判结果的优劣，这点还应该继续去发掘探索。

八、参考资料

1. GIZA++源码: <https://github.com/moses-smt/giza-pp>
2. GIZA++解析: <https://www.docin.com/p-922474358.html>
3. 使用 GIZA 进行词对齐的博客:
<https://blog.csdn.net/csdnofzyk/article/details/82683065>
4. GIZA++运行报告:
(1)
https://wenku.baidu.com/view/511580da844769eae109ed98.html?rec_flag=default
(2)
https://blog.csdn.net/tianliang0123/article/details/5301847?spm=1001.2101.3001.6650.6&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-6.no_search_link&depth=1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7Edefault-6.no_search_link
5. 卡内基梅隆大学的 grow-diag-final-and 方法:
<http://www.cs.cmu.edu/afs/cs/project/cmt-55/1ti/Courses/731/www/Spring-11/Slides/MT-Class-110202-PhraseAlign.ppt>

九、附录

1. 附录 1

```
import os

filepathen = './en'
filelisten = os.listdir(filepathen)
filelisten = sorted(filelisten)
print(filelisten)
```

```

filepathcn = './cn'
filelistcn = os.listdir(filepathcn)
filelistcn = sorted(filelistcn)
print(filelistcn)

print(len(filelistcn), len(filelisten))
cnfile = ""
for filename in filelistcn:
    with open('./cn/' + filename) as f:
        for line in f:
            cnfile += line
with open("cnfile.txt", "w") as f:
    f.write(cnfile)

enfile = ""
for filename in filelisten:
    with open('./en/' + filename) as f:
        for line in f:
            enfile += line
with open("enfile.txt", "w") as f:
    f.write(enfile)

```

2. 附录 2

```

from itertools import izip
import re
from sys import argv, exit

# 0 - include alignments to 'NULL' token in GROW-DIAG-FINAL
# 1 - exclude alignments to 'NULL' token in GROW-DIAG-FINAL
START_INDEX = 1

def init_matrix(rows, columns, value):
    """
    A function that returns matrix in a given
    dimensions filled with a given value
    """

    result = [[value for _ in range(columns)] for _ in range(rows)]
    return result

def intersection(e2f, f2e):
    rows = len(e2f)

```

```

columns = len(f2e)
result = init_matrix(rows, columns, False)
for e in range(rows):
    for f in range(columns):
        result[e][f] = e2f[e][f] and f2e[f][e]

return result

def union(e2f, f2e):
    rows = len(e2f)
    columns = len(f2e)
    result = init_matrix(rows, columns, False)
    for e in range(rows):
        for f in range(columns):
            result[e][f] = e2f[e][f] or f2e[f][e]

    return result

def neighboring_points((e_index, f_index), e_len, f_len):
    """
    A function that returns list of neighboring points in
    an alignment matrix for a given alignment (pair of indexes)
    """
    result = []

    if e_index > 0:
        result.append((e_index - 1, f_index))
    if f_index > 0:
        result.append((e_index, f_index - 1))
    if e_index < e_len - 1:
        result.append((e_index + 1, f_index))
    if f_index < f_len - 1:
        result.append((e_index, f_index + 1))
    if e_index > 0 and f_index > 0:
        result.append((e_index - 1, f_index - 1))
    if e_index > 0 and f_index < f_len - 1:
        result.append((e_index - 1, f_index + 1))
    if e_index < e_len - 1 and f_index > 0:
        result.append((e_index + 1, f_index - 1))
    if e_index < e_len - 1 and f_index < f_len - 1:
        result.append((e_index + 1, f_index + 1))

```

```

    return result

def aligned_e(e, f_len, alignment):
    """
    A function that checks if a given 'english' word is aligned
    to any foreign word in a given foreign sentence
    """
    for f in range(START_INDEX, f_len):
        if alignment[e][f]:
            return True

    return False

def aligned_f(f, e_len, alignment):
    """
    A function that checks if a given foreign word is aligned
    to any 'english' word in a given 'english' sentence
    """
    for e in range(START_INDEX, e_len):
        if alignment[e][f]:
            return True

    return False

def grow_diag(union, alignment, e_len, f_len):
    new_points_added = True
    while new_points_added:
        new_points_added = False
        for e in range(START_INDEX, e_len):
            for f in range(START_INDEX, f_len):
                if alignment[e][f]:
                    for (e_new, f_new) in neighboring_points((e, f), e_len, f_len):
                        if not (aligned_e(e_new, f_len, alignment) and aligned_f(f_new, e_len,
alignment)) \
                            and union[e_new][f_new]:
                            alignment[e_new][f_new] = True
                            new_points_added = True

def final(alignment, e2f, f2e, e_len, f_len):
    """

```

```

A function that implements both FINAL(e2f) and FINAL(f2e)
steps of GROW-DIAG-FINAL algorithm
"""

for e in range(START_INDEX, e_len):
    for f in range(START_INDEX, f_len):
        if not (aligned_e(e, f_len, alignment) and aligned_f(f, e_len, alignment)) \
            and (e2f[e][f] or f2e[f][e]):
            alignment[e][f] = True

def final_e2f(alignment, e2f, e_len, f_len):
    """
    A function that implements FINAL(e2f) step of GROW-DIAG-FINAL algorithm
    """
    for e in range(START_INDEX, e_len):
        for f in range(START_INDEX, f_len):
            if not (aligned_e(e, f_len, alignment) and aligned_f(f, e_len, alignment)) \
                and e2f[e][f]:
                alignment[e][f] = True

def final_f2e(alignment, f2e, e_len, f_len):
    """
    A function that implements FINAL(f2e) step of GROW-DIAG-FINAL algorithm
    """
    for e in range(START_INDEX, e_len):
        for f in range(START_INDEX, f_len):
            if not (aligned_e(e, f_len, alignment) and aligned_f(f, e_len, alignment)) \
                and f2e[f][e]:
                alignment[e][f] = True

def grow_diag_final(e2f, f2e, e_len, f_len):
    alignment = intersection(e2f, f2e)
    grow_diag(union(e2f, f2e), alignment, e_len, f_len)
    final(alignment, e2f, f2e, e_len, f_len)
    return alignment

def parse_alignments	alignments_line, values):
    word_alignments_regex = ur"(\S+)\s\(\{([s\d]*)\}\)"
    alignments = re.findall(word_alignments_regex, alignments_line)

    # Initialize matrix with False value for each pair of words

```



```

rows = len(alignments)
columns = len(values)
result = init_matrix(rows, columns, False)

# Align words
for i in range(len(alignments)):
    alignment_values = alignments[i][1].split()
    for alignment in alignment_values:
        result[i][int(alignment)] = True

return result

def print_alignments(alignments, e_len, f_len):
    result = ''
    for f in range(1, f_len):
        for e in range(1, e_len):
            if alignments[e][f]:
                result += str(f) + '-' + str(e) + ' '

    print result.strip()

def main():
    if len(argv) < 3:
        exit('Usage: %s e2f_alignments_file f2e_alignments_file' % argv[0])

    script, e2f_filename, f2e_filename = argv

    # States:
    # 0 - skip line with information about sentences length and alignment score
    # 1 - read sentences
    # 2 - read alignments and run GROW-DIAG-FINAL
    state = 0

    e_sentence = []
    f_sentence = []

    with open(e2f_filename) as e2f_file, open(f2e_filename) as f2e_file:
        for e2f_line, f2e_line in izip(e2f_file, f2e_file):
            if state == 0:
                state = 1
            elif state == 1:
                f_sentence = ['NULL'] + e2f_line.split()

```

```

        e_sentence = ['NULL'] + f2e_line.split()
        state = 2
    elif state == 2:
        alignments = grow_diag_final(
            parse_alignments(e2f_line, f_sentence),
            parse_alignments(f2e_line, e_sentence),
            len(e_sentence), len(f_sentence))
        print_alignments(alignments, len(e_sentence), len(f_sentence))
        break

if __name__ == '__main__':
    main()

```

3. 附录 3

```

# coding=utf-8
import codecs
from sys import argv, exit

import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

START_INDEX = 1

def main():
    if len(argv) < 5:
        exit('Usage: %s e_token_file f_token_file aligned_file line_to_plot' % argv[0])

    _, e_token_file, f_token_file, aligned_file, line_to_plot = argv

    # Prepare data
    e_sentence = ''
    f_sentence = ''
    point_indices = ''
    with codecs.open(e_token_file, encoding='utf-8') as f:
        for idx, line in enumerate(f):
            if idx == int(line_to_plot):
                e_sentence = line
                break

```

```

with codecs.open(f_token_file, encoding='utf-8') as f:
    for idx, line in enumerate(f):
        if idx == int(line_to_plot):
            f_sentence = line
            break
with open(aligned_file) as f:
    for idx, line in enumerate(f):
        if idx == int(line_to_plot):
            points = line.strip().split()
            if START_INDEX:
                points = map(lambda p: (int(p[0]) - 1, int(p[1]) - 1), [point.split('-')
for point in points])
            else:
                points = map(lambda p: (int(p[0]), int(p[1])), [point.split('-') for point
in points])

            point_indices = list(zip(*points))
            break

x_ticks = e_sentence.split()
y_ticks = f_sentence.split()

align_matrix = np.zeros(shape=(len(y_ticks), len(x_ticks)))
align_matrix[point_indices] = 1

# Plot
fig, ax = plt.subplots()

plt.imshow(align_matrix, cmap='binary')
plt.xticks(np.arange(len(x_ticks)), x_ticks, rotation=90)
plt.yticks(np.arange(len(y_ticks)), y_ticks)

ax.set_xticks(np.arange(len(x_ticks)) + 0.5, minor=True) # Grid lines
ax.set_yticks(np.arange(len(y_ticks)) + 0.5, minor=True)
ax.xaxis.grid(True, which='minor')
ax.yaxis.grid(True, which='minor')

ax.xaxis.set_ticks_position('top')
plt.savefig('static/out.jpg')
plt.show()

if __name__ == '__main__':
    main()

```

