# 大作业二-汉英词语自动对齐系统

顾桉丞 1120201760
李锦宇 1120201230
宋奕骁 1120201795
刘勇奇 1120201334

2023 年 1 月 9 日

## 目录

## 一、核心思想和算法描述

我们实现的是汉英词语自动对齐系统。

核心思想：参照了在分词算法中较为流行的 GIZA++ 的源码作为蒙版，进行进一步的改写与移植来实现我们的平台。然后在 GUI 中实现用户与程序的交互。

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 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，也就是 en.txt 和 zh.txt 文件。

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 都是字典文件，en_zh.snt / zh_en.snt 中编号表示句对，第一行表示句对出现次数，

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
```

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
```

en.vcb.classes / zh.vcb.classes 是单词所属类别编号，en.vcb.classes.cats / zh.vcb.classes.cats 是类别所拥有的一组单词。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 -CoocurrenceFile
    zh_en.cooc -o z2e -OutputPath z2e
./GIZA++ -S en.vcb -T zh.vcb -C en_zh.snt -CoocurrenceFile
    en_zh.cooc -o e2z -OutputPath e2z
```

输出的文件中，z2e.perp 是困惑度。

z2e.a3.final 文件里有 i j l m p(i/j, l, m)，其中 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。

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

z2e.n3.final 中 source_id $p_0 p_1 p_2 \dots p_n$；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。

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。

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

z2e.d4.final 是 IBM Model 4 翻译表。

z2e.D4.final 是 IBM Model 4 的 Distortion 表。

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，以上的文件详见附录，接下来便是系统主要模块流程。

## 二、系统主要模块流程

词性标注系统主要分为四部分：初始化，调用可执行文件，分析结果和可视化。接下来我们逐个去进行流程分析。

### （一）初始化-exchange.py

exchange.py 的作用是将从网上下载的 LDC 语料库转化成便于读取的 txt 文件。从开放数据平台 LDC-宾夕法尼亚大学官网下载的 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。该语料库下载之后会提供两个文件夹，里面对应存储的是中文语句和对应翻译的英文语句，我们的 exhcnage.py 程序便是将文件夹中的中英文翻译读取，排好序后写入 enfile.txt 和 cnfile.txt 两个文件。

首先需要将存储在 cn，en 文件夹中的中英文语句的文件名列表，并排好序，以保证中文和英文是句句对应的。

```
    '''
    处理中文英文语料库，读入以进行后续处理
    :return:两个文件
    '''
    fileenlist  = os.listdir("./en")
    fileenlist  = sorted( fileenlist )

    filezhlist  = os.listdir("./cn")
    filezhlist  = sorted( filezhlist )
```

接下来，我们将列表中的文件信息都读取出来，并写入我们准备好的 cnfile.txt 和 enfile.txt。

```
    zhfile = ""
    '''
    读取语料库并写入文件
    '''
    for filepath in filezhlist :
        filetruepath = str('./cn/' + filepath)
        with open(filetruepath) as f:
            for line in f:
                zhfile += line
    with open("cnfile.txt", "w") as f:
```

```
        f.write( zhfile )

    enfile = ""
    for filepath in fileenlist :
        filetruepath = str('./en/' + filepath)
        with open(filetruepath) as f:
            for line in f:
                enfile += line
    with open("enfile.txt", "w") as f:
        f.write( enfile )
```

**（二）调用可执行文件**-diaoyong.py

diaoyong.py 的主要功能为调用编译 GIZA 原码获得的四个可执行文件：mkcls、plain2snt.out、snt2cooc.out、GIZA++，将获得的最终结果输出到 z2e 和 e2z 两个文件夹中并且删除中间文件。

首先，我们接受用户输入的中文语句和对应的英文翻译。当然，用户输入的中文语句一般不会进行分词。所以，我们选择写一个基于统计的分词程序，该分词程序采取的语料库为人民日报语料库。下面给大家展示一下核心的 Viterbi 算法。

```
        def viterbi(wordnet):
            dis = []
            for i in range(len(wordnet)):
                dis.append(dict())
            node = []
            for i in range(len(wordnet)):
                node.append(dict())
            word_line = []
            for i in range(len(wordnet)):
                word_line.append(dict())
            wordnet[len(wordnet) - 1].append("#末末#")
            # 更新第一行
            for word in wordnet[1]:
                dis [1][ word] = ca_weight(bi_grams, "#始始#"
```

```
                    , word, ca_gram_num(bi_grams))
15              node[1][word] = 0
16              word_line[1][word] = "#始始#"
17
18          # 遍历wordnet的每一行
19          for i in range(1, len(wordnet) - 1):
20              for word in wordnet[i]:
21                  # 更新加上这个单词的距离之后那个位置的
                        所有单词的距离
22                  for to in wordnet[i + len(word)]:
23                      if word in dis[i]:
24                          if to in dis[i + len(word)]:
25                              # 只要最大频率
26                              if dis[i + len(word)][to] < dis
                                    [i][word] * ca_weight(
                                    bi_grams, word, to,
                                    bi_grams_num):
27                                  dis[i + len(word)][to] =
                                        dis[i][word] *
                                        ca_weight(bi_grams,
                                        word, to,
                                        bi_grams_num)
28                                  node[i + len(word)][to] = i
29                                  word_line[i + len(word)][to
                                        ] = word
30                          else:
31                              dis[i + len(word)][to] = dis[i
                                    ][word] * ca_weight(
                                    bi_grams, word, to,
                                    bi_grams_num)
32                              node[i + len(word)][to] = i
33                              word_line[i + len(word)][to] =
                                    word
```

```
34
35              # 回溯
36              path = []
37              f = node[len(node) - 1]["#末末#"]
38              fword = word_line[len(word_line) - 1]["#末末#"]
39              path.append(fword)
40              while f:
41                  tmpword = fword
42                  fword = word_line[f][tmpword]
43                  f = node[f][tmpword]
44                  path.append(fword)
45              path = path[:-1]
46              path.reverse()
47              return dis, node, word_line, path
48
49          si_net = gen_wordnet(si_grams, text)
50          (dis, _, _, path) = viterbi(si_net)
51
52          return path
```

在对输入的中文语句分词之后，我们将中英文语句保存在对应的训练集当中。

```
1  def main(enfiles, cnfiles):
2
3      '''
4      此处为读取文字信息，将文本格式进行处理，得到双语语
           料，之后调用GIZA++程序，实现中英文的文本对齐
5      GIZA++程序为四个可执行文件，在本处使用linux命令耐用
           执行，分别为plain2snt.out，snt2cooc.out、GIZA++、
           mkcls
6      调用main函数来启动程序
7      '''
8
```

```
9       enfilepath = "enfile.txt"
10      cnfilepath = "cnfile.txt"
11      # 调用fenxi模块，进行词性标注
12
13      cnfiles = fenci.main(cnfiles)
14
15      # 读取英文文件到txten中
16      with open(enfilepath, "r") as f:
17          txten = f.readlines()
18          f.close()
19
20      # 将读出的英文末尾加换行，同时将其反转并组装成新的
            txtens写入源文件
21      with open(enfilepath, "w") as f:
22          enfiles += "\n"
23          txten.append(enfiles)
24          txten.reverse()
25          txtens = ""
26          for item in txten:
27              txtens += item
28          f.write(txtens)
29          f.close()
30
31      # 读取中文文件到txtcn中
32      with open(cnfilepath, "r") as f:
33          txtcn = f.readlines()
34          f.close()
35      # 将读出的中文末尾加换行，同时将其反转并组装成新的
            txtcns写入源文件
36      with open(cnfilepath, "w") as f:
37          cnfiles += "\n"
38          txtcn.append(cnfiles)
39          txtcn.reverse()
```

```
40          txtcns = ""
41          for item in txtcn:
42              txtcns += item
43          f.write(txtcns)
44          f.close()
```

在准备好训练集之后，我们通过 python 中的 os.system() 方法来调用 GIZA 的可执行文件。首先，我们调用命令进行词对齐：

```
1     # 执行plain2snt.out程序，参数为中英文文本所在路径，作用
          为为中英文词编号
2     # 得到enfile.vcb、cnfile.vcb、cnfile_enfile.snt、
          enfile_cnfile.snt四个文件
3     os.system("./plain2snt.out " + cnfilepath + " " +
          enfilepath)
```

在进行词对齐之后，我们会得到对应的中间文件。在之后，我们生成共现文件：

```
1     # 执行snt2cooc.out程序，生成中英文共现文件
2     # 输入为cnfile.vcb enfile.vcb cnfile_enfile.snt，输出到
          cnfile_enfile.cooc文件中，下行同理
3     os.system("./snt2cooc.out cnfile.vcb enfile.vcb
          cnfile_enfile.snt > cnfile_enfile.cooc")
4     os.system("./snt2cooc.out enfile.vcb cnfile.vcb
          enfile_cnfile.snt > enfile_cnfile.cooc")
```

生成共现文件之后，我们进一步调用可执行文件，构建词类：

```
1     # 生成中英文词类 opt为优化输出的命令
2     # .vcb.classes文件表示单词所属类别编号
3     os.system("./mkcls -pcnfile.txt -Vcnfile.vcb.classes opt")
4     os.system("./mkcls -penfile.txt -Venfile.vcb.classes opt")
```

在构建词类之后，我们需要创建两个文件夹 z2e 和 e2z 用来存放输出结果，最后我们将结果输出到两个文件夹中：

```
    # 建立中英文对应的输出文件夹，否则没有输出文件夹GIZA
        ++会报错
    if not os.path.exists("z2e"):
        os.mkdir("z2e")
    if not os.path.exists("e2z"):
        os.mkdir("e2z")

    # 执行GIZA++的执行文件，使中英文本对齐
    # 参数:
    # -o 文件前缀
    # -OutputPath 输出所有文件到文件夹
    # 最终的输出文件存储在e2z和z2e两个文件夹中
    os.system(
        "./GIZA++ -S cnfile.vcb -T enfile.vcb -C
            cnfile_enfile.snt -CoocurrenceFile cnfile_enfile.cooc
             -o z2e -OutputPath z2e")
    os.system(
        "./GIZA++ -S enfile.vcb -T cnfile.vcb -C
            enfile_cnfile.snt -CoocurrenceFile enfile_cnfile.cooc
             -o e2z -OutputPath e2z")
```

最后，我们需要将产生的中间文件全部删除：

```
    os.system("rm enfile.vcb")
    os.system("rm cnfile.vcb")
    os.system("rm enfile.vcb.classes")
    os.system("rm enfile.vcb.classes.cats")
    os.system("rm cnfile.vcb.classes")
    os.system("rm cnfile.vcb.classes.cats")
    os.system("rm enfile_cnfile.cooc")
    os.system("rm enfile_cnfile.snt")
    os.system("rm cnfile_enfile.cooc")
    os.system("rm cnfile_enfile.snt")
```

### （三）分析结果-fenxi.py

fenxi.py 的功能是将 z2e 中的单向对齐文件结果 z2e.A3.final 读取出来，并且根据其结果将最后的图绘制出来并保存在当前目录下的 out.jpg。

首先，我们需要将单向对齐文件中的数据取出来，并且将数据保存 cheng

```
# 打开单向对齐文件，数字代表Token所在句子位置（1为起点）
with open("z2e/z2e.A3.final") as f:
    txt = f.readlines()
    f.close()
# print(txt)

# 获取英文数据
# I like natural language classes \n
endata = txt[1].split()
# 获取中文数据
# NULL ({ }) 我 ({ 1 }) 喜欢 ({ 2 }) 自然 ({ 3 }) 语言 ({ 4 }) 课程 ({ 5 }) \n
dqdata = txt[2].split(")")
for item in range(len(dqdata)):
    dqdata[item] = dqdata[item].split("(")

# 存放对齐结果的字典，键是中文，值对应的位置
dqdatas = {}

# 截取对齐数据
dqdata = dqdata[:-1]

# 将对齐数据转换为字典
for item in dqdata:
    item[1] = item[1][1:-1].strip()
    item[1] = item[1].split()
    item[0] = item[0].strip()
```

```
27            dqdatas[item[0]] = item[1]
```

接下来，将分词结果分解成坐标，得到最后的散点图和分词、对齐结果。

```
1     # 截取中文分词结果
2     cndata = list(dqdatas.keys())[1:]
3
4     # 存放坐标
5     tu = list()
6     # 点的横坐标
7     tx = []
8     # 点的纵坐标
9     ty = []
10
11    # 获取点即中英文位置对应关系
12    for i in range(len(cndata)):
13        cndatas = list(dqdatas[cndata[i]])
14        for item in cndatas:
15            tu.append((i + 1, eval(item)))
16            tx.append(i + 1)
17            ty.append(eval(item))
18
19    cnstr = ""      # 中文分词结果
20    for item in cndata:
21        cnstr += item
22        cnstr += " "
23
24    # 插入两坐标轴原点
25    endata.insert(0, "英文")
26    cndata.insert(0, "中文")
27
28    # 设置图片字体
29    my_font = font_manager.FontProperties(fname=(r'微软雅
          黑.ttf'))
```

```
30
31      # 设置横纵坐标，设置图片标题
32      plt.scatter(tx, ty)
33      plt.xticks(range(len(cndata)), cndata, fontproperties=
            my_font)
34      plt.yticks(range(len(endata)), endata, fontproperties=
            my_font)
35      plt.title("最终结果", fontproperties=my_font)
36
37      # 保存图片
38      plt.savefig("./out.jpg")
39
40      #清除当前图片
41      plt.clf()
42
43      fcstr = ""    # 对齐结果
44      for item in tu:
45          fcstr += str(item)
46          fcstr += " "
47
48      # 返回中文分词结果和对齐结果
49      return cnstr, fcstr
```

**（四）可视化**-Maintest.py+GUI.py

Maintest.py 将上述三个模块进行串联，让其可以统一执行，便于 GUI 进行控制。

```
1       def main(enfile, cnfile):
2           '''
3           调用函数，执行测试检测执行效果
4           :param enfile:英文语料库
5           :param cnfile:中文语料库
6           :return:fcstr , dqstr
7           '''
```

```
    if not os.path.exists("cnfile.txt") or not os.path.exists("
        enfile.txt"):
        exchange.main()

    diaoyong.main(enfile, cnfile)

    [fcstr, dqstr] = fenxi.main()
    return fcstr, dqstr
```

GUI.py 是基于 Tkinter 库编写的 GUI 可视化代码。在本部分，我们只给出相关的事件函数，完整代码将在附录里面放出。

```
    def event(self):
        if not self.YXZTLabel["text"] == "正在运行中……":
            self.YXZTLabel["text"] = "正在运行中……"

        eninput = self.enet.get()
        cninput = self.cnet.get()
        try:
            [fcstr, dqstr] = Maintest.main(eninput, cninput)
            self.fcjgLabel["text"] = fcstr
            self.dqjgLabel["text"] = dqstr
            photo = ImageTk.PhotoImage(Image.open('out.jpg')
                .resize((320, 240)))
            self.ImageLabel = tk.Label(self.root, image=photo)
            self.ImageLabel.image = photo
            self.ImageLabel.place(x=350, y=80)
        except:
            self.YXZTLabel["text"] = "运行错误！"
        else:
            self.YXZTLabel["text"] = "运行结束"
```

## 三、实验结果与分析

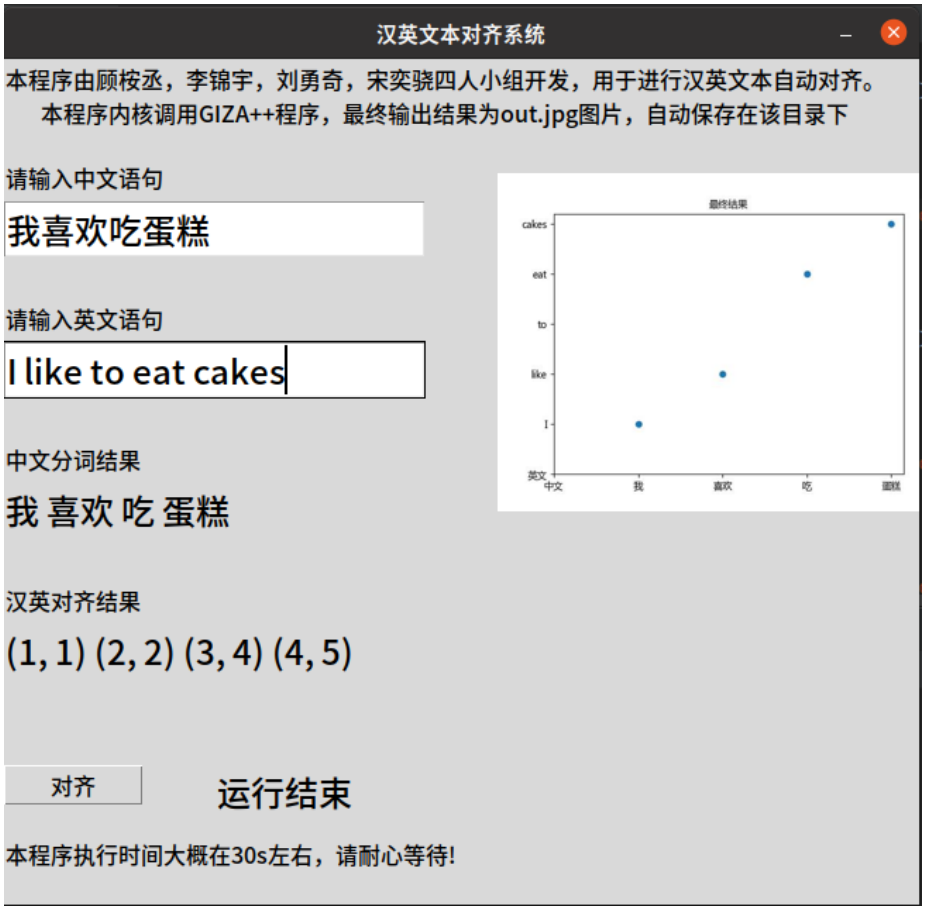


图 1: 结果 1

我们的 GUI 也可以多次运行结果，图 2 为第二次运行的结果：


图 2: 结果 2

由上图的结果，中文语句为“我喜欢北京理工大学”，对应的分词结果为“我 喜欢 北京 理工大学”。根据这个运行结果可知，该运行程序并不能完全分析出一些专有名词，而“我喜欢吃蛋糕”这一用例则运行的相对比较完美，推测是由于语料库的时间局限性导致。同时英文语句的匹配出现了一对多的情况，说明这个程序能够进行一些初步的简单的分析，但是准确率依然不足以达到预期水平。初步推测后续可以通过语料库的扩充，以及算法迭代的更新，实现更高的分词准确率。

## 四、总结展望与反思

### （一）实验开发总结

本系统以中英文对齐的工具包 GIZA++ 为基础，使用 viterbi 算法对用户的中文输入进行分词和预处理，最终使用 Tkinter 作为 GUI 开发模块开发用户交互界面，基本实现了分词功能与中英文对齐功能。并能对用户的输入进行处理和响应。

除了核心的 GIZA++ 工具模块，我们设计和实现了自己的基于统计分词的分词程序，对文本内容进行对应和切分的预处理程序以及后期对于对齐结果的可视化输出和用户的交互程序，以得到合理的，可靠性强的分词结果和良好的交互体验。

### （二）实验的反思与展望

首先是语料库的局限性，本实验的语料库选取较为单一，对于不同环境下的分词和识别能力较弱，使得该分词系统对于日常语言运用场合效果对齐效果较好，对于实体名词较多，语言规范性较差的文本识别效果并不理想。

以语料库为着眼点的优化可以考虑扩大语料库，增加其他场景的语言资料。另外可以将系统根据语料库来源划分使用场景，提高不同场景下的分词与对齐准确率。

其次是选用模型的局限性。GIZA++ 实现的算法较早，经过二十余年的发展，尤其是近 10 年来从统计机器翻译 (SMT) 到神经网络机器翻译 (NMT) 的跨越, 深度学习算法在准确率上存在明显优势，通过更换使用的算法和工具，可以进一步的提升系统的准确度。

## 五、实验人员和分工情况

小组成员共 4 人：李锦宇，顾桉丞，宋奕骁，刘勇奇。

选题和设计由小组成员商讨决定，模块实现过程主要由李锦宇和顾桉丞负责，系统的整合，调试与最终运行结果记录由刘勇奇和宋奕骁完成。实验报告通过 overleaf 线上编辑共同完成。

## 六、参考文献

[1] 俞敬松, 王惠临, 吴胜兰. 高正确率的双语语块对齐算法研究 [J]. 中文信息学报,2015,29(01):67-74+110.

[2] 刘小虎, 吴葳, 李生, 赵铁军, 蔡萌, 鞠英杰. 基于词典和统计的语料

库词汇级对齐算法 [J]. 情报学报,1997(01):20-26.

[3] 邓丹, 刘群, 俞鸿魁. 基于双语词典的汉英词语对齐算法研究 [J]. 计算机工程,2005(16):45-47.

## 七、附录

zh.txt:

```
1 海洋 是 一个 非常 复杂 的 事物 。
2 人类 的 健康 也 是 一 件 非常 复杂 的 事情 。
3 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 。
4 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 。
5 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:

```
1 It can be a very complicated thing , the ocean .
2 And it can be a very complicated thing , what human health is .
3 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 .
4 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 .
5 And I 'm going to start with this one : If momma ain 't happy
    , ain 't nobody happy .
```

en.vcb / zh.vcb:

```
1 2 海洋 1
2 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：

```
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
 ...
```

en.vcb.classes / zh.vcb.classes：

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

en.vcb.classes.cats / zh.vcb.classes.cats：

```
0:$,
1:
2:science,
```

```
3:seem,
4:things,
5:some,
6:start,
7:task,
```

z2e.perp:

| #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 | |

```
14 5 0 12 Model3 8.93033 N/A 9.50349 N/A
15 5 0 13 Model3 7.68766 N/A 8.19622 N/A
16 5 0 14 Model3 6.64154 N/A 7.04977 N/A
17 5 0 15 T3To4 6.17993 N/A 6.55567 N/A
18 5 0 16 Model4 6.16858 N/A 6.4715 N/A
19 5 0 17 Model4 6.0819 N/A 6.39317 N/A
20 5 0 18 Model4 6.04302 N/A 6.34387 N/A
21 5 0 19 Model4 5.95066 N/A 6.2234 N/A
```

z2e.a3.final：

```
1 1 8 100 1
2 5 2 8 100 1
3 4 3 8 100 1
4 2 4 8 100 1
5 5 5 8 100 1
6 4 6 8 100 1
7 4 7 8 100 1
8 0 8 8 100 1
9 ...
```

z2e.d3.final：

```
1 2 0 100 8 0.0491948
2 6 0 100 8 0.950805
3 3 1 100 8 1
4 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:

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

```
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:

```
# 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：

```
# 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：

```
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
...
```

GUI 部分的完整代码：

```
    class GUI:

    def ___init___(self):
        '''
        规定窗口基本格式：大小650×600，题目为“汉英文本对齐系统”
        '''
        self.root = tk.Tk()
        self.root.title('汉英文本对齐系统')
        self.root.geometry("650x600")
        self.root.resizable(width=False, height=False)
        self.interface()

    def interface(self):
        '''
        本段规定gui界面类的各个属性，包括操作标签，介绍，输入框以及提示
        :return:
        '''
        self.btn = tk.Button(self.root, text="对齐", font=("fangsong ti", 12), command=self.event)
        self.btn.place(x=0, y=500, width= 100, height= 30)
```

```
20
21          self.ZTLabel = tk.Label(self.root, font=("song ti", 12)
                , anchor='w')
22          self.ZTLabel["text"] = "本程序由顾桉丞，李锦宇，刘
                勇奇，宋奕骁四人小组开发，用于进行汉英文本自动
                对齐。\n本程序内核调用GIZA++程序，最终输出结
                果为out.jpg图片，自动保存在该目录下"
23          self.ZTLabel.place(x=0, y=0)
24
25          self.cnLabel = tk.Label(self.root, font=("song ti", 12)
                , anchor='w')
26          self.cnLabel["text"] = "请输入中文语句"
27          self.cnLabel.place(x=0, y=80)
28
29          self.cnet = tk.StringVar()
30          self.cnentry = tk.Entry(self.root, font=("fangsong ti",
                18), textvariable=self.cnet)
31          self.cnentry.place(x=0, y=100, width=300)
32
33          self.enLabel = tk.Label(self.root, font=("song ti", 12)
                , anchor='w')
34          self.enLabel["text"] = "请输入英文语句"
35          self.enLabel.place(x=0, y=180)
36
37          self.enet = tk.StringVar()
38          self.enentry = tk.Entry(self.root, font=("fangsong ti",
                18), textvariable=self.enet)
39          self.enentry.place(x=0, y=200, width=300)
40
41          self.fcLabel = tk.Label(self.root, font=("song ti", 12)
                , anchor='w')
42          self.fcLabel["text"] = "中文分词结果"
43          self.fcLabel.place(x=0, y=280)
```

```
44
45          self.fcjgLabel = tk.Label(self.root, font=("fangsong ti
                ", 18), anchor='w')
46          self.fcjgLabel["text"] = ""
47          self.fcjgLabel.place(x=0, y=300)
48
49          self.dqLabel = tk.Label(self.root, font=("song ti", 12)
                , anchor='w')
50          self.dqLabel["text"] = "汉英对齐结果"
51          self.dqLabel.place(x=0, y=380)
52
53          self.dqjgLabel = tk.Label(self.root, font=("fangsong ti
                ", 18), anchor='w')
54          self.dqjgLabel["text"] = ""
55          self.dqjgLabel.place(x=0, y=400)
56
57          self.ZTendLabel = tk.Label(self.root, font=("song ti",
                12), anchor='w')
58          self.ZTendLabel["text"] = "本程序执行时间大概在30s左
                右，请耐心等待!"
59          self.ZTendLabel.place(x=0, y=550)
60
61      def event(self):
62          eninput = self.enet.get()
63          cninput = self.cnet.get()
64          [fcstr, dqstr] = Maintest.main(eninput, cninput)
65          self.fcjgLabel["text"] = fcstr
66          self.dqjgLabel["text"] = dqstr
67          photo = ImageTk.PhotoImage(Image.open('out.jpg').
                resize((320, 240)))
68          self.ImageLabel = tk.Label(self.root, image= photo)
69          self.ImageLabel.image = photo
70          self.ImageLabel.place(x=350, y=80)
```

```
71
72 if __name__ == '__main__':
73     a = GUI()
74     a.root.mainloop()
```