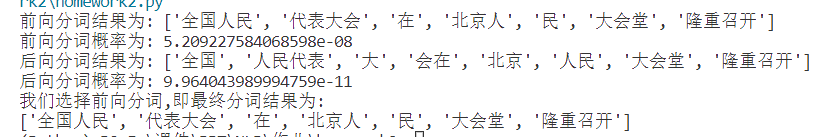
一、第二次作业源码：

1. **import** re
2. **import** jieba
3. **import**  numpy as np
4. **import** copy
5. #构建数据库
6. wordset={}
7. diction={}
8. **def** train02(filename):
9. f=open(filename,encoding='utf-8')
10. #一共有多少个句子
11. num=0
12. **for** line **in** f.readlines():
13. # 匹配所有中文形成一个list包,按符号分割出所有的字符
14. line1 = re.findall('[\u4e00-\u9fa5]+', line)
15. **for** words **in** line1:
16. #存前一个字符
17. words1='B'+words+'E'
18. pre=' '
19. words1List=jieba.lcut(words1)
20. #计算一个词的频数和两个词连续出现的频数
21. **for** word **in** words1List:
22. **for** word **in** words1List:
23. **if** word **in** diction.keys():
24. diction[word] += 1
25. **else**:
26. diction[word] = 1
27. **if** pre != ' ':
28. **if** pre **in** wordset:
29. **if** word **in** wordset[pre]:
30. wordset[pre][word]+=1
31. **else**:
32. wordset[pre][word]=1
33. **else**:
34. wordset[pre] = {}
35. wordset[pre][word] = 1
36. pre=word
37. num+=1
38. f.close()
39. np.save("./data/wordset.npy", wordset)
40. np.save("./data/dict.npy", diction)
42. **class** ChineseTokenizer:
43. **def** \_\_init\_\_(self,dictPath,diction,wordset):
44. """
45. 中文分词器，包括FMM与BMM算法
47. """
48. #统计词的词典
49. self.dictionary = self.createDictionary(dictPath)
50. #统计一个词出现的频率
51. self.diction=diction
52. #统计两个词一起出现的频率
53. self.wordset=wordset
55. #创造统计词的词典
56. **def** createDictionary(self,dictPath):
57. Dictonary = list()
58. **for** line **in** open(dictPath,encoding='utf-8'):
59. Dictonary += line.strip().split(' ')
60. **return** Dictonary
62. **def** FMM(self, sentence):
63. """
64. 正向最大匹配算法，正向移动，缩短子串时删除末尾的字
65. param sentence: 待分词句子
66. return res: 分词列表
67. """
68. res = []
69. # 外层循环切句子
70. **while** len(sentence) > 0:
71. # 初始化一个切词划分窗口，取词典最长词长度和句长的最小值
72. split\_win\_len = min(4, len(sentence))
73. # 初始化词
74. sub\_sen = sentence[0:split\_win\_len] # 正向移动
75. # 内层循环匹配词
76. **while** len(sub\_sen) > 0:
77. # 如果词在词典中
78. **if** sub\_sen **in** self.dictionary:
79. res.append(sub\_sen)
80. **break**
81. # 如果词长度为1，说明词典中没有当前词
82. **elif** len(sub\_sen) == 1:
83. res.append(sub\_sen)
84. **break**
85. # 如果当前词在词典中没有，那么删除末尾的字，再进入while匹配
86. **else**:
87. split\_win\_len -= 1 # 划分窗口缩短
88. sub\_sen = sub\_sen[0:split\_win\_len] # 子串尾部pop
89. # 一次匹配结束更新原句子
90. sentence = sentence[split\_win\_len:]
91. **return** res

94. **def** BMM(self, sentence):
95. """
96. 逆向最大匹配算法，逆向移动，缩短子串时删除首部的字
98. param :
99. return:
100. """
101. res = []
102. **while** len(sentence) > 0:
103. split\_win\_len = min(4, len(sentence))
104. sub\_sen = sentence[-split\_win\_len:]
105. **while** len(sub\_sen) > 0:
106. **if** sub\_sen **in** self.dictionary: # 逆向移动
107. res.append(sub\_sen)
108. **break**
109. **elif** len(sub\_sen) == 1:
110. res.append(sub\_sen)
111. **break**
112. **else**:
113. split\_win\_len -= 1 # 划分窗口缩短
114. sub\_sen = sub\_sen[-split\_win\_len:] # 子串首部pop
115. sentence = sentence[0:-split\_win\_len]
116. res = res[::-1]
117. **return** res
119. #查找wordset
120. **def** findWordset(self,worda,wordb):
121. **if** worda **in** self.wordset:
122. **if** wordb **in** self.wordset[worda]:
123. **return** self.wordset[worda][wordb]
124. **else**:
125. **return** 0
126. **else**:
127. **return** 0
129. #查找wordset中某一个的例子
130. **def** findWordset1(self,word):
131. **if** word **in** self.wordset:
132. **return** len(self.wordset[word])
133. **else**:
134. **return** 1
136. #查找diction
137. **def** findDiction(self,word):
138. **if** word **in** self.diction:
139. **return** self.diction[word]
140. **else**:
141. **return** 0
143. #计算得分，用拟二元文法模型
145. **def** pingfen(self,Res):
146. Result=copy.deepcopy(Res)
147. P = 1
148. Result.append('E')
149. Result.insert(0,'B')
150. **for** i **in** range(1,len(Result)):
151. res=1.0\*(self.findWordset(Result[i-1],Result[i])+1)/(self.findDiction(Result[i-1])+self.findWordset1(Result[i-1]))
152. P=P\*res
153. **return** P


157. **if** \_\_name\_\_=='\_\_main\_\_':
158. #train02('C:/Users/19033/Desktop/PPT/NLP/作业/homework2/data/news2.txt')
159. diction = np.load('F:/课件/PPT/NLP/作业/homework2/data/dict1.npy', allow\_pickle='TRUE').item()
160. wordset = np.load('F:/课件/PPT/NLP/作业/homework2/data/wordset.npy', allow\_pickle='TRUE').item()
161. #print(diction)
162. #print(wordset)
163. model=ChineseTokenizer('F:/课件/PPT/NLP/作业/homework2/data/dict.txt',diction,wordset)
164. sentences="全国人民代表大会在北京人民大会堂隆重召开"
165. Fmmlist=model.FMM(sentences)
166. Bmmlist=model.BMM(sentences)
167. FmmlistPingfen=model.pingfen(Fmmlist)
168. BmmlistPingfen=model.pingfen(Bmmlist)
169. **print**("前向分词结果为:",Fmmlist)
170. **print**("前向分词概率为:",FmmlistPingfen)
171. **print**("后向分词结果为:",Bmmlist)
172. **print**("后向分词概率为:",BmmlistPingfen)
173. #消除歧义
174. **if**(FmmlistPingfen>BmmlistPingfen):
175. **print**("我们选择前向分词,即最终分词结果为:")
176. **print**(Fmmlist)
177. **else**:
178. **print**("我们选择后向分词,即最终分词结果为:")
179. **print**(Bmmlist)
181. 运行结果：



具体运行结果如上，我们运用了二元文法模型来进行歧义的消除，进一步提高分词的精度，但可能是数据库本身的问题，导致爬取的结果不是很好。