题目：基于训练语料，训练一个基于字的Bigram语言模型。当用户输入某个字序列，程序可以自动推荐该序列的后一个字（依次列出概率最大的5个可能字选项），根据提示用户选择某个字后，程序可以继续推荐下一个字的列表。例如：输入“长江大”，程序猜测下一个可能的字为“桥”、“河”、“学”、“道”等。

**思路：**

1. 首先我们需要用语料库构建我们的训练模型。第一步我们需要构建词到词性的映射，以便于后面进行分词，然后我们为每一个句子添加“BOS“和”EOS”，这样保证了条件概率在i=1的时候有意义，也保证了句子内所有字符串的和概率为1。

**def Pretreatment（）:语料预处理**

def Pretreatment(file\_path):  
 intab = "０１２３４５６７８９"  
 outtab = "0123456789"  
 trantab = str.maketrans(intab, outtab) # 将所有的in字符与对应的out字符建立映射  
 f1 = open("filter.txt",  
 "w", encoding='utf-8') # 构造过滤后的文件  
 for line in open(file\_path, encoding='gbk'):  
 new\_line = re.sub(  
 r" |/t|/n|/m|/v|/u|/a|/w|/q|r|t|/k|/f|/p|n|/c|s|/|d|i|b|l|j|e|v|g|N|V|R|T|y|o|A|D|h|z|x|A|B|M|a|Y|\d{8}-\d{2}-\d{3}-\d{3}", "", line)  
 new\_line = new\_line.translate(trantab) # 将所有的in字符用对应的out字符替代  
 f1.write(new\_line)  
 f1.close()

**preDiv（）：传入训练语料，返回添加了“BOS“和”EOS”的训练语料**

def preDiv(filename):  
 listSen = []  
 with open(filename, 'r', encoding='UTF-8') as f:  
 for line in f.readlines():  
 listTmp = list(jieba.cut(line))  
 listSen.append("BOS")  
 listSen.extend(listTmp)  
 listSen.append("EOS")  
 f.close()  
 return listSen

1. 然后我们统计语料库的词频，返回一个词语和出现次数对应的字典和该字典的大小（即语料库里词语的总数）

**Statistic()**：传入已经做好分词了的句子库，然后统计词频，返回词频对应字典和字典大小。

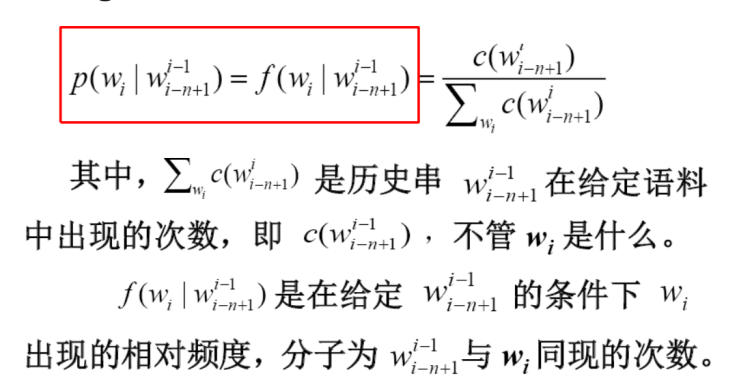
def statistic(lists):  
 dictSen = {}  
 for word in lists:  
 if word not in punc:  
 if word not in dictSen:  
 dictSen[word] = 1  
 else:  
 dictSen[word] += 1  
 dictSen = sorted(dictSen.items(), key=lambda x: x[1], reverse=False)  
 dictSen = dict(dictSen)  
 dictLen = len(dictSen)  
 return dictSen, dictLen

1. 对于要预测的句子，首先我们要对句子进行分词，这里我们调用了jieba这个包，应用里面的jieba.cut（）函数进行分词，同时注意的是我们忽略了标点符号。

**CutWord（）:传入句子，返回分词后的句子**

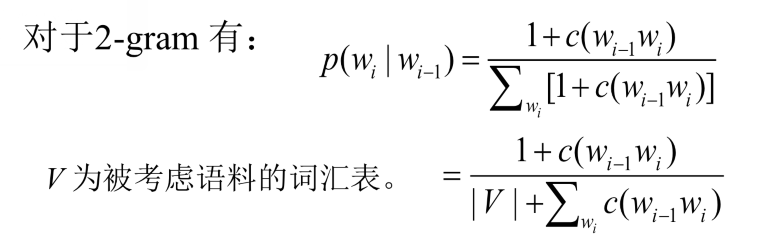
def cutWord(sen):  
 cutResult = list(jieba.cut(sen))  
 tmp = []  
 for word in cutResult:  
 if word not in punc:  
 tmp.append(word)  
 cutResult = tmp  
 cutResult.insert(0, "BOS")  
 cutResult.append("EOS")  
 print("分词结果为：{}\n".format(cutResult))  
 return cutResult

1. 对于要预测的句子，我们使用二元文法来预测他可能出现的下一个字。
2. 根据公式我们知道，首先我们需要算出p，我们设计两个列表，fenzi和fenmu，他们的第k个数分别代表了计算第k个词语p的分子分母。然后我们遍历该句子，对于第i个词，要计算他的p,就需要找到它前一个词出现的次数，即从词频统计字典里找到第i-1个词出现的次数。然后找到在i-1个词出现的情况下，第i个词出现的次数，即从之前的句子库里找出相关次数。



1. 然后我们找到待预测句子里的最后一个字，然后去词频字典里找到所有第一个字为该字的词语，将其可能出现的下一个字和频率放入一个新的字典里面。然后再对字典排序，找出前五个可能的下一个字。
2. 然后我们计算出预测出来的句子的概率，这里我们对新加入的字采用1）中的方法来计算相应的p
3. 对于那些在本次数据库里出现概率为0的词语，我们采用了加一平滑法，

对于可能概率为0的词，我们采用了加一平滑法，即如下式：



然后我们用第i个词的对应分子除以分母算出p,然后相乘即可求出此次预测句子的概率。

**bigram()：传入词频统计字典、字典大小、已经分词了的句子。**

def biGram(listSen, dicSen, dicLen, cutResult):  
 fenzi = [0]\*(len(cutResult)-1)  
 fenmu = []  
 for i in range(1, len(cutResult)):  
 # 计算分母  
 if cutResult[i-1] in dicSen:  
 fenmu.append(dicSen[cutResult[i-1]])  
 else:  
 fenmu.append(0)  
 # 计算分子  
 for j in range(len(listSen)):  
 if (listSen[j] == cutResult[i-1]) and listSen[j+1] == cutResult[i]:  
 fenzi[i-1] += 1  
  
 end = len(cutResult) - 1;  
 timesDict = {}  
 for key, value in dicSen.items():  
 if key[0] == cutResult[end - 1]:  
 if len(key)==2:  
 timesDict[key[1]] = value  
  
 newfenmu = fenmu  
 newfenzi = fenzi  
 timesDict = sorted(timesDict.items(), key = lambda kv:(kv[1], kv[0]),reverse=True)  
 count = 0  
 for word in timesDict:  
 key = word[0]  
 count+=1  
 if count == 6:  
 break  
 k = 1  
 b = k \* (dicLen+1)  
  
 if key in dicSen:  
 newfenmu.append(dicSen[key])  
 else:  
 newfenmu.append(0)  
 newfenzi.append(1)  
 for j in range(len(listSen) - 1):  
 if (listSen[j] == cutResult[end]) and listSen[j + 1] == key:  
 newfenzi[end+1] += 1  
 # 使用加法数据平滑  
 newfenzi = [i + k for i in newfenzi]  
 newfenmu = [i + b for i in newfenmu]  
  
 print("下一个字可能为 %s" % key)  
 result = 1.0  
 for s in range(len(newfenzi)):  
 if newfenmu[s] != 0:  
 tmp = newfenzi[s] / newfenmu[s]  
 result \*= tmp  
 print("概率为 {}\n".format(result))

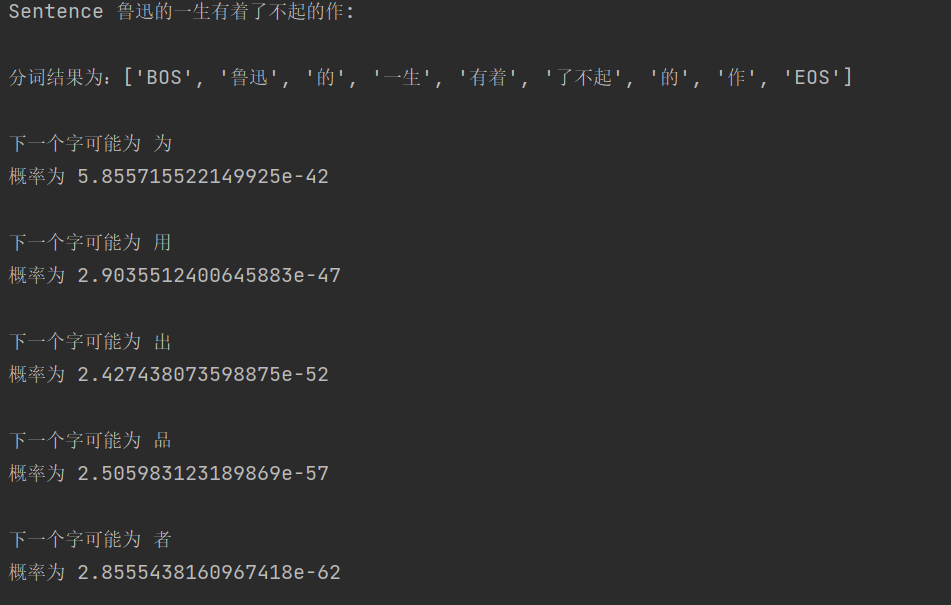
1. **main函数**

这里我们首先对语料预处理，然后用三个句子进行测试。

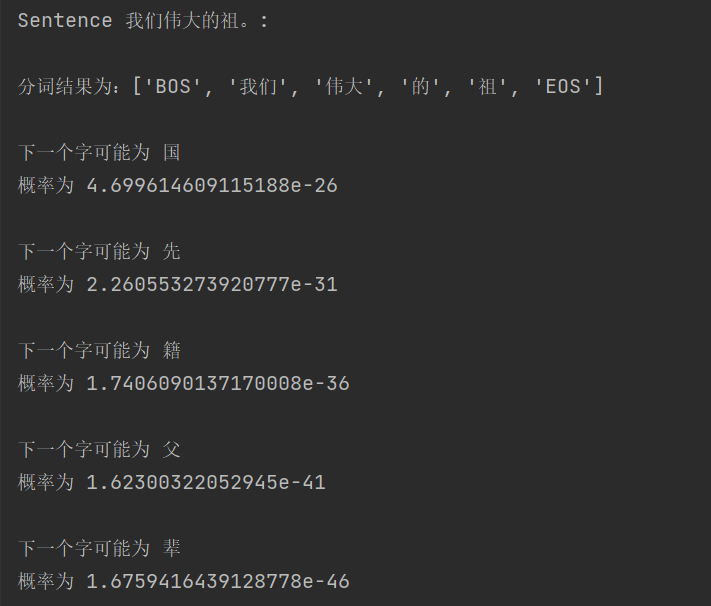
if \_\_name\_\_ == "\_\_main\_\_":  
 Pretreatment("训练语料.txt")  
 filename = "filter.txt"  
 listSen = preDiv(filename)  
 sen1 = "鲁迅的一生有着了不起的作"  
 sen3 = "我们伟大的祖。"  
 sen4 = "坚持中国共产党的领"  
 testData = [sen1, sen3, sen4]  
 cnt = 0 # 计数  
 dictSen, dictLen = statistic(listSen)  
 for ss in testData:  
 print("Sentence {}: \n".format(ss))  
 cnt += 1  
 cutResult = cutWord(ss)  
 biGram(listSen, dictSen, dictLen, cutResult)

**实验结果：**

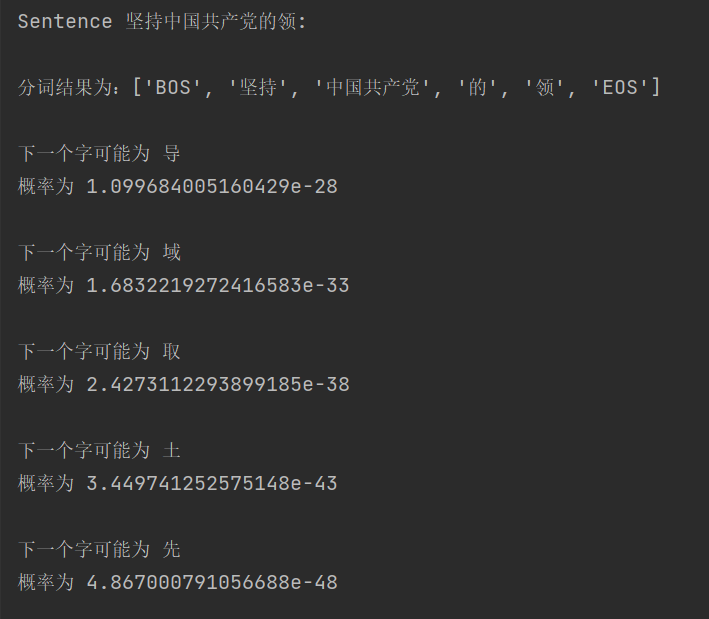
1. 句子一：鲁迅的一生有着了不起的作 理想下一个字应该为：”为”,



1. 句子二：我们伟大的祖 ， 理想的下一个字为”国“。



1. 句子三：坚持中国共产党的领 ，理想的下一个字为”导“。



附：整个代码

import jieba  
import re  
  
  
# 待替代符号  
punc = "\n ！？｡＂＃《》＄％＆＇（）＊＋－／：；＜＝＞＠［＼］＾＿｀｛｜｝～｟｠｢｣〃》「」『』【】〔〕〖〗〘〙〚〛〜〝〞〟〰〾〿–—‘’‛“”„‟…‧﹏#$%&'()\*+-/;<=>?@[\]^\_`{|}~“”？！【】（）。’‘……￥"  
  
  
def Pretreatment(file\_path):  
 intab = "０１２３４５６７８９"  
 outtab = "0123456789"  
 trantab = str.maketrans(intab, outtab) # 将所有的in字符与对应的out字符建立映射  
 f1 = open("filter.txt",  
 "w", encoding='utf-8') # 构造过滤后的文件  
 for line in open(file\_path, encoding='gbk'):  
 new\_line = re.sub(  
 r" |/t|/n|/m|/v|/u|/a|/w|/q|r|t|/k|/f|/p|n|/c|s|/|d|i|b|l|j|e|v|g|N|V|R|T|y|o|A|D|h|z|x|A|B|M|a|Y|\d{8}-\d{2}-\d{3}-\d{3}", "", line)  
 new\_line = new\_line.translate(trantab) # 将所有的in字符用对应的out字符替代  
 f1.write(new\_line)  
 f1.close()  
  
  
def preDiv(filename):  
 listSen = []  
 with open(filename, 'r', encoding='UTF-8') as f:  
 for line in f.readlines():  
 listTmp = list(jieba.cut(line))  
 listSen.append("BOS")  
 listSen.extend(listTmp)  
 listSen.append("EOS")  
 f.close()  
 return listSen  
  
  
def statistic(lists):  
 dictSen = {}  
 for word in lists:  
 if word not in punc:  
 if word not in dictSen:  
 dictSen[word] = 1  
 else:  
 dictSen[word] += 1  
 dictSen = sorted(dictSen.items(), key=lambda x: x[1], reverse=False)  
 dictSen = dict(dictSen)  
 dictLen = len(dictSen)  
 return dictSen, dictLen  
  
  
def cutWord(sen):  
 cutResult = list(jieba.cut(sen))  
 tmp = []  
 for word in cutResult:  
 if word not in punc:  
 tmp.append(word)  
 cutResult = tmp  
 cutResult.insert(0, "BOS")  
 cutResult.append("EOS")  
 print("分词结果为：{}\n".format(cutResult))  
 return cutResult  
  
  
def biGram(listSen, dicSen, dicLen, cutResult):  
 fenzi = [0]\*(len(cutResult)-1)  
 fenmu = []  
 for i in range(1, len(cutResult)):  
 # 计算分母  
 if cutResult[i-1] in dicSen:  
 fenmu.append(dicSen[cutResult[i-1]])  
 else:  
 fenmu.append(0)  
 # 计算分子  
 for j in range(len(listSen)):  
 if (listSen[j] == cutResult[i-1]) and listSen[j+1] == cutResult[i]:  
 fenzi[i-1] += 1  
  
 end = len(cutResult) - 1;  
 timesDict = {}  
 for key, value in dicSen.items():  
 if key[0] == cutResult[end - 1]:  
 if len(key)==2:  
 timesDict[key[1]] = value  
  
 newfenmu = fenmu  
 newfenzi = fenzi  
 timesDict = sorted(timesDict.items(), key = lambda kv:(kv[1], kv[0]),reverse=True)  
 count = 0  
 for word in timesDict:  
 key = word[0]  
 count+=1  
 if count == 6:  
 break  
 k = 1  
 b = k \* (dicLen+1)  
  
 if key in dicSen:  
 newfenmu.append(dicSen[key])  
 else:  
 newfenmu.append(0)  
 newfenzi.append(1)  
 for j in range(len(listSen) - 1):  
 if (listSen[j] == cutResult[end]) and listSen[j + 1] == key:  
 newfenzi[end+1] += 1  
 # 使用加法数据平滑  
 newfenzi = [i + k for i in newfenzi]  
 newfenmu = [i + b for i in newfenmu]  
  
 print("下一个字可能为 %s" % key)  
 result = 1.0  
 for s in range(len(newfenzi)):  
 if newfenmu[s] != 0:  
 tmp = newfenzi[s] / newfenmu[s]  
 result \*= tmp  
 print("概率为 {}\n".format(result))  
 # print(result)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 Pretreatment("训练语料.txt")  
 filename = "filter.txt"  
 listSen = preDiv(filename)  
 sen1 = "鲁迅的一生有着了不起的作"  
 sen3 = "我们伟大的祖。"  
 sen4 = "坚持中国共产党的领"  
 testData = [sen1, sen3, sen4]  
 cnt = 0 # 计数  
 dictSen, dictLen = statistic(listSen)  
 for ss in testData:  
 print("Sentence {}: \n".format(ss))  
 cnt += 1  
 cutResult = cutWord(ss)  
 biGram(listSen, dictSen, dictLen, cutResult)