

《自然语言处理》实验报告

一、实验目的

1. 理解、掌握二元文法，并将其应用于实际情景；
2. 学会使用数据平滑处理语料。

二、实验项目内容

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。

要求：

- （1）要求至少使用一种平滑方法。
- （2）提交电子文档一份（word），内含两部分内容：程序源码文本，程序运行结果截图（至少包含三个测试语句运行结果）；
- （3）同时提交源程序文件（可采用任意语言开发）。

训练语料可以用实验课提供给大家的 SogouLabDic。

三、实验过程或算法（源程序）

1. 二元文法语言模型描述

计算语句 $s=w_1w_2...w_m$ 的先验概率时，经常使用公式：

$$p(s) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1w_2) \times \dots \times p(w_m|w_1...w_{m-1})$$
$$= \prod_{i=1}^m p(w_i|w_1...w_{i-1})$$

w_i 称为统计基元，可以是字、词、短语或词类。 w_i 的概率取决于 $w_1w_2...w_{i-1}$ ，即 w_i 的历史。

为了便于统计和运算，只考虑有限长度的历史，通常假设当前词只依赖前 $N-1$ 个词，称为 N 元文法。本实验采用二元文法，即当前词只依赖前一个词，形成马尔科夫链。因此上述公式可简化为：

$$p(s) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_2) \times \dots \times p(w_m|w_{m-1})$$
$$= \prod_{i=1}^m p(w_i|w_{i-1})$$

为了保证概率条件在 $i=1$ 时有意义，在句首加入<BOS>。但本次实验并不需要统计初始情况的概率分布，因为初始汉字是键盘输入的。

2. 拉普拉斯平滑描述

为了避免零概率问题，会对统计得到的每个 N 元对的次数作调整。通常采用“劫富济贫”的思想，增大出现次数较少的 N 元对的比例，减少出现次数较多的 N 元对的比例。这种“劫富济贫”的思想称为数据平滑，常见的数据平滑分为加 1 法、减值法和删除插值法等。由于后两种方法在一些实现过程中需要处理超参数，较为麻烦，因此此处采用第一种数据平滑方法来调整 N 元对。

加 1 法也称为拉普拉斯平滑，基本思想就是将 N 元文法模型中每个 N 元对的出现次数加 1。这样就可以将原本出现次数为 0 的 N 元对变成出现 1 次，从而避免了零概率的情况。

对于二元文法，有以下平滑公式：

$$p(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i} [1 + c(w_{i-1}w_i)]}$$
$$= \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

3. 实验过程


```

#语料分句
N=len(hanzi_list)
trans_list=[[0]*N for i in range(N)]
sentence=[] #语料库,每个元素为一句不含标点的字的汉字列表
with open(r'NLP\data\news.txt','r',encoding='UTF-8-sig') as f:
    while True:
        line=f.readline()
        if not line: #读取结束
            break
        line=line.replace('\n','') #此处line是字符串
        tmp=re.findall('[\u4e00-\u9fa5]+',line)
        if len(tmp): #对于空行或无中文字符的行不予添加
            sentence.extend(tmp)

```

3.3 统计转移概率

使用上一步处理过的中文分句，提取其中的二元词组并作统计：

```

#统计二元词频
for x in sentence:
    for i in range(len(x)):
        if i==0: #语料的第一个字直接跳过
            continue
        else:
            #x[i-1]->x[i]
            if (x[i-1] not in hanzi_list) or (x[i] not in hanzi_list):
                continue
            else:
                trans_list[hanzi_index[x[i-1]]][hanzi_index[x[i]]]+=1

```

3.4 计算转移概率+数据平滑

根据统计的二元词组，计算转移概率并加入数据平滑：

```

#加一法进行数据平滑
for i in range(N):
    sum=0
    for j in range(N):
        sum+=(trans_list[i][j]+1)
    for j in range(N):
        trans_list[i][j]=(trans_list[i][j]+1)/sum

```

3.5 循环读入并预测结果

使用while循环，不断读入输入的汉字，并预测其后的5个最有可能的汉字。对于不符合规范的输入，采取必要的措施结束程序的运行：

```

if __name__ == '__main__':
    trans_list,hanzi_list,hanzi_index=Construct_Dict()
    N=len(trans_list)
    sentence=''
    print("Please input a Chinese character:")
    while(True):
        word=str(input())
        #提取输入的汉字并作判断
        chn=re.findall('[\u4e00-\u9fa5]',word)
        if len(chn)==0: #输入不符合规则
            print('Unallowed input!')
            break
        if word not in hanzi_list: #字典无法识别输入汉字或输入字数超过1
            print('Unkowned Chinese character!')
            print(sentence,end=' ')
            continue
        sentence+=word
        #预测下一个字
        tmp=trans_list[hanzi_index[word]].copy() #tmp存储当前汉字转移到下一个汉字的概率
        lst=[] #lst存储下一个汉字和其对应的概率
        for i in range(len(tmp)):
            st={}
            st['char']=hanzi_list[i]
            st['p']=tmp[i]
            lst.append(st)
        top5=heapq.nlargest(5,lst,lambda x:x['p'])
        for i in range(5):
            print(top5[i]['char'],end=' ')
        print("\n"+sentence,end=' ')

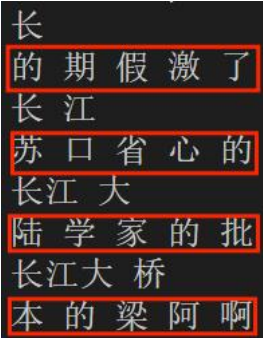
```

其中推荐最有可能的下一个字时，先将当前字的所有转移概率取出，将其与各汉字配对，放入列表。再使用`heapq.nlargest`函数对概率进行堆排序，最终得到5个概率最大的推荐字。

四、实验结果及分析和（或）源程序调试过程

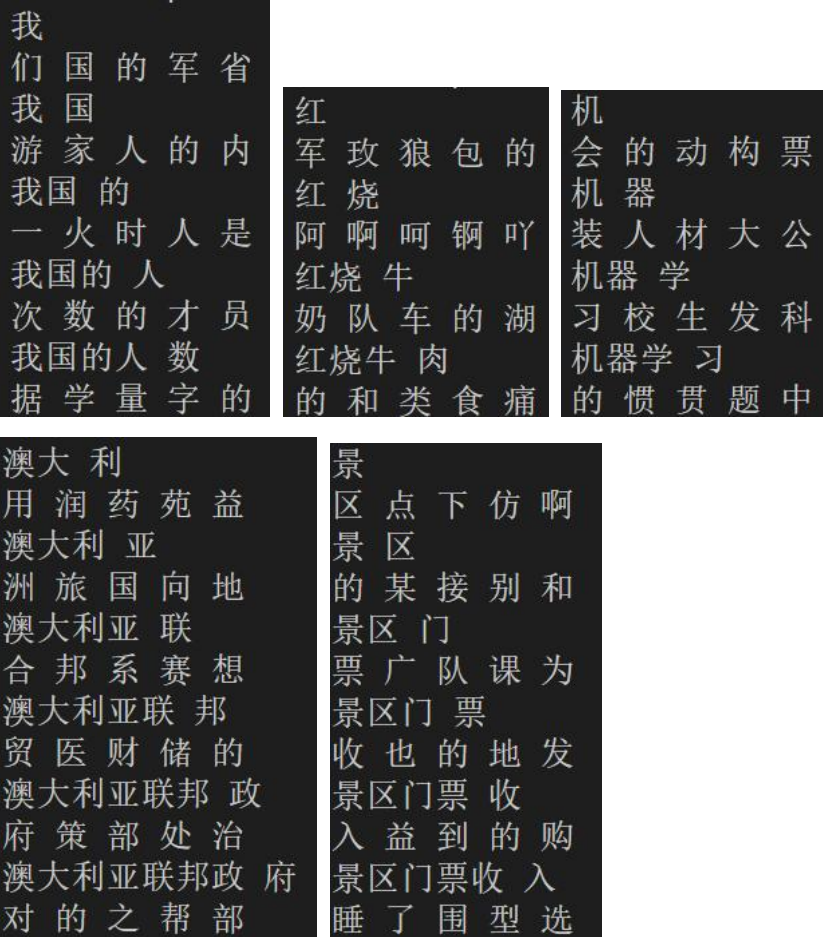
实验结果：

如图所示，每输入一个字，就可以推荐 5 个概率最大的汉字：



长
的 期 假 激 了
长 江
苏 口 省 心 的
长 江 大
陆 学 家 的 批
长 江 大 桥
本 的 梁 阿 啊

图中圈出的是推荐的字，其余是输入的字。更多样例如下：



我	红	机
们 国 的 军 省	军 玫 狼 包 的	会 的 动 构 票
我 国	红 烧	机 器
游 家 人 的 内	阿 啊 呵 钢 吖	装 人 材 大 公
我 国 的	红 烧 牛	机 器 学
一 火 时 人 是	奶 队 车 的 湖	习 校 生 发 科
我 国 的 人	红 烧 牛 肉	机 器 学 习
次 数 的 才 员	的 和 类 食 痛	的 惯 贯 题 中
我 国 的 人 数		
据 学 量 字 的		

澳 大 利	景
用 润 药 苑 益	区 点 下 仿 啊
澳 大 利 亚	景 区
洲 旅 国 向 地	的 某 接 别 和
澳 大 利 亚 联	景 区 门
合 邦 系 赛 想	票 广 队 课 为
澳 大 利 亚 联 邦	景 区 门 票
贸 医 财 储 的	收 也 的 地 发
澳 大 利 亚 联 邦 政	景 区 门 票 收
府 策 部 处 治	入 益 到 的 购
澳 大 利 亚 联 邦 政 府	景 区 门 票 收 入
对 的 之 帮 部	睡 了 围 型 选

可以看到，模型对于常见的词组推荐较为准确，对于两个词间的衔接有待提高。对于异常输入，处理情况如下：

对于输入非汉字的字符，程序将提示非法输入并直接停止运行：

```
我
们 国 的 军 省
我 I
Unallowed input!
```

对于输入超过一个汉字或者字典中不存在的汉字，程序将提示并允许用户重新输入：

```
长
的 期 假 激 了
长 江
苏 口 省 心 的
长江 大桥
Unknowed Chinese character!
长江 大
陆 学 家 的 批
```

调试过程：

1. 问题：正则表达式匹配汉字时只能匹配单个汉字；
解决方法：在表达式后加上“+”，允许匹配多个汉字：

```
tmp=re.findall('[\u4e00-\u9fa5]+',line)
```

2. 问题：提取语料中有空语句，对处理造成麻烦；
解决方法：因为语料格式不规范，存在一些空行或者整行中无中文字符，因此统计前判断是否是空语句：

```
if len(tmp): #对于空行或无中文字符的行不予添加
    sentence.extend(tmp)
```

3. 问题：读取 news.txt 时，语料中出现的标点以及英文无法在字典中匹配；
解决方法：使用正则表达式匹配所有汉字。

实验感悟：

二元文法模型较为简单，在推荐一个完整的二元词语时比较有效。但对于一句话的推荐，涉及不同词语间的连接，甚至是不同词性间的转换，二元文法的效果显然不尽人意。

有两种办法可以使得推荐结果更加令人满意：

- 使用三元甚至是四元文法，使得推荐字可以更加依赖历史，推荐出的字也会更加合理；
- 增大训练集，使得模型能够识别更多词汇与语义。

五、附录（完整代码）

```
'''基于字的 Bigram 语言模型

    功能要求: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。要求至少使用一种平滑方法。

    训练集: 'NLP\data\news.txt'

    分析:此模型不同于实验 1 的 HMM,已经不存在隐状态,直接统计词频即可

    思路:统计词频->循环推荐汉字

'''
import re
import heapq

def Construct_Dict():
    '''读取 pinyin2hanzi.txt 文件,得到所有汉字的列表+汉字在列表中的索引
    读取 news.txt 文件,统计词频,计算转移概率,加入数据平滑,返回概率矩阵
    return:
        trans_list:汉字之间的转移概率
        hanzi_list:汉字的列表
        hanzi_index:汉字在列表中的索引
    '''
    #统计汉字
    hanzi_list=[]                #汉字的列表
    hanzi_index={}              #汉字在列表中的索引
    i=0
    with open(r'NLP\data\pinyin2hanzi.txt','r',encoding='UTF-8-sig') as f:
        while True:
            line=f.readline()
            if not line:         #读取结束
                break
            line=line.replace('\n','') #去掉结尾换行符
            line=line.split(' ')      #分割为拼音和汉字
            for x in line[1]:
                if x not in hanzi_list: #x 是第一次出现的汉字
                    hanzi_list.append(x)
                    hanzi_index[x]=i
                    i+=1
```



```

#语料分句
N=len(hanzi_list)
trans_list=[[0]*N for i in range(N)]
sentence=[] #语料库,每个元素为一句不含标点的字的汉字列表
with open(r'NLP\data\news.txt','r',encoding='UTF-8-sig') as f:
    while True:
        line=f.readline()
        if not line: #读取结束
            break
        line=line.replace('\n','') #此处 line 是字符串
        tmp=re.findall('[\u4e00-\u9fa5]+',line)
        if len(tmp): #对于空行或无中文字符的行不予添加
            sentence.extend(tmp)

#统计二元词频
for x in sentence:
    for i in range(len(x)):
        if i==0: #语料的第一个字直接跳过
            continue
        else:
            #x[i-1]->x[i]
            if (x[i-1] not in hanzi_list)or(x[i] not in hanzi_list):
                continue
            else:
                trans_list[hanzi_index[x[i-1]]][hanzi_index[x[i]]]+=1

#加一法进行数据平滑
for i in range(N):
    sum=0
    for j in range(N):
        sum+=(trans_list[i][j]+1)
    for j in range(N):
        trans_list[i][j]=(trans_list[i][j]+1)/sum

return trans_list,hanzi_list,hanzi_index

if __name__ == '__main__':
    trans_list,hanzi_list,hanzi_index=Construct_Dict()
    N=len(trans_list)

```

```

sentence=''
print("Please input a Chinese character:")
while(True):
    word=str(input())
        #提取输入的汉字并作判断
    chn=re.findall('[\u4e00-\u9fa5]',word)
    if len(chn)==0:                #输入不符合规则
        print('Unallowed input!')
        break
    if word not in hanzi_list:      #字典无法识别输入汉字或输入字数超过 1
        print('Unknowed Chinese character!')
        print(sentence,end=' ')
        continue
    sentence+=word
    #预测下一个字
    tmp=trans_list[hanzi_index[word]].copy()    #tmp 存储当前汉字转移到下一个汉
字的概率
    lst=[]                                       #lst 存储下一个汉字和其对应的概率
    for i in range(len(tmp)):
        st={}
        st['char']=hanzi_list[i]
        st['p']=tmp[i]
        lst.append(st)
    top5=heapq.nlargest(5,lst,lambda x:x['p'])
    for i in range(5):
        print(top5[i]['char'],end=' ')
    print("\n"+sentence,end=' ')

```