# 《自然语言处理》实验报告

## 一、实验目的

1. 理解、掌握隐马尔可夫模型，N 元语法等自然语言处理的基本思想、算法；

2. 将其应用于从汉语拼音到汉字的自动转换过程。

假定：拼音串中已经用空格进行了分隔，如“wo ai wo jia”

## 二、实验项目内容

1. 对训练语料及相关资源进行预处理；
2. 通过学习算法，训练 HMM 模型；
3. 利用 HMM 模型和维特比算法，实现从任意拼音到汉字的自动转换。
4. 利用给定测试集，评价上述程序的转换准确率。

## 三、实验过程或算法（源程序）

### 1. HMM 模型描述

本实验用到隐马尔科夫模型 HMM，该模型含有一个隐状态序列 $Q_i$，观察输出序列 $O_j$，隐状态序列间存在一个状态转移概率矩阵 A，隐状态到观察状态存在一个发射概率矩阵 B。实验者根据观察到的观察序列的值，求得该观察序列最有可能对应的隐状态序列。

实验中，建立拼音转汉字程序，隐状态为汉字，观察到的输出为拼音，要解决的问题是通过观察到的拼音序列预测最有可能的汉字序列。具体符号声明如下：

| 符号 | 描述 |
|---|---|
| Q | 隐状态的集合（汉字） |
| O | 观察状态的集合（拼音） |
| $A=\{a_{ij}\}$ | 状态转移概率矩阵 |
| $B=\{b_{jk}\}$ | 发射概率矩阵 |
| $\pi=\{p(q_1=S_i)\}$ | 初始状态概率分布 |

记该 HMM 模型为$\mu=(A, B, \pi)$

### 2. Viterbi 搜索算法描述

在已知 HMM 模型$\mu=(A, B, \pi)$后，可以利用动态规划求解概率最大的对应观察序列的隐状态序列。记$\delta_t(i)$表示时间 t 时隐状态为 Si 且观察序列为

报告创建时间：2022.10.7

$O_1O_2...O_t$ 的最大概率，$\Psi_t(j)$表示$\delta_t(i)$对应的前驱节点。

（1）初始化：

$$\delta 1(i) = \pi_i * b_i(O_1)$$
$$\Psi 1(j) = 0$$

（2）递推关系：

$$\delta_t(i) = \max_{1<=i<=N}(\delta_{t-1}(i) * a_{ij}) * b_j(O_t)$$

$$\Psi_t(j) = \operatorname{argmax}_{i}(\delta_{t-1}(i) * a_{ij}) * b_j(O_t)$$

（3）终止条件：

$$q_T = \operatorname{argmax}_{1<=i<=N}\delta_T(i)$$

$$p(q_T) = \max_{1<=i<=N}\delta_T(i)$$

（4）回溯路径：

$$q_t = \Psi_{t+1}(q_{t+1})$$

**3. 实验过程**

实验流程为：读取数据—>训练 HMM 模型—>实现 Viterbi 搜索算法—>测试模型正确率

3.1 数据预处理

读取 pinyin2hanzi.txt 文件，得到所有汉字（即隐状态）对应的拼音（观察结果），从而使得后面计算发射概率矩阵 B 时能够识别汉字的读音。该文件数据格式为：

```
a 阿啊呵锕吖腌嗄
ai 爱嫒暧瑷哀锿挨埃诶唉隘嗌艾哎砹癌蔼霭矮碍皑
```

因此每次读取一行，**去掉结尾换行符**后用空格分割拼音和汉字。由于 HMM 模型和 Viterbi 搜索算法在使用隐状态和观测状态时都是按编号索引，因此需要将拼音与汉字存储到列表并用字典记录每个汉字和拼音对应的下标。

并且为了处理一字多音和一音多字，还记录了同一个字对应的不同读音和同一个读音对应的不同字。

```python
def Construct_Dict():
    '''读取pinyin2hanzi.txt文件,构造隐显状态集合,即汉字和拼音的列表并记录其下标
        return:
            hanzi_list:隐状态的列表
            pinyin_list:显状态的列表
            hanzi_index:隐状态对应的hanzi_list中坐标的字典
            pinyin_index:显状态对应的pinyin_list中坐标的字典
            hanzi2pinyin_dic:隐状态可能对应的显状态集合的字典
            pinyin_dic:显状态可能对应的隐状态集合的字典
    '''
    hanzi_list=[]
    pinyin_list=[]
    hanzi_index={}
    pinyin_index={}
    hanzi2pinyin_dic={}
    pinyin2hanzi_dic={}
    i=0
    j=0
    with open(r'NLP\实验1\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
            pinyin_list.append(line[0])
            pinyin_index[line[0]]=i
            i+=1
            pinyin2hanzi_dic[line[0]]=set()
            for x in line[1]:
                pinyin2hanzi_dic[line[0]].add(x)
                if x not in hanzi_list: #x是第一次出现的汉字
                    hanzi_list.append(x)
                    hanzi_index[x]=j
                    j+=1
                    # hanzi2pinyin_dic[x]=set(line[0])  #字符串line[0]会被拆成多个字符
                    hanzi2pinyin_dic[x]=set()
                    hanzi2pinyin_dic[x].add(line[0])
                else:                   #x是多音字
                    hanzi2pinyin_dic[x].add(line[0])
    return hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic
```

### 3.2 HMM 模型训练

读取 toutiao_cat_data.txt 文件，训练得到状态转移概率矩阵 A 和初始状态矩阵π。该文件数据格式为：

```
6551700932705387022_!_101_!_news_culture_!_京城最值得你来场文化之旅的博物馆_!_保利集团,马未都,中国科学技术馆,博物馆,新中国
6552368441838272771_!_101_!_news_culture_!_发酵床的垫料种类有哪些？哪种更好？_!_
```

每行为一条数据，以“_!_”分割的个字段，从前往后分别是新闻 ID，分类 code，分类名称，新闻标题，新闻关键词（有的词条不含关键词），详见中文文本分类数据集。训练**只需要用到新闻标题**，因此每读取一行用“_!_”分割并提取标题。此外，新闻标题中还存在空格以及标点符号，需要使用**正则表达式将汉字提取**出来。

对于读取到的语料中可能出现未标记的汉字，选择直接丢弃。出现过的汉字即可标记其状态转移的次数以及初始状态分布，最终计算出 A 和π。

对于发射概率矩阵 B，由于语料库的局限性，toutiao_cat_data.txt 文件中的汉字并未标注拼音，因此发射概率只能用最低级的均分获得。

由于训练时间较长，第一次训练结束后便将结果（矩阵 A、B、Pi）写入 json 文件，后续使用时就不必再每次花时间训练即可直接使用。

```
def Train_miu(hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic):
    '''读取toutiao_cat_data.txt文件,构造状态转移概率矩阵A,发射概率矩阵B,初始状态概率矩阵Pi
        return: A,B,Pi
    '''
    N=len(hanzi_list)
    M=len(pinyin_list)
    A=[[0]*N for i in range(N)]
    B=[[0]*M for i in range(N)]
    Pi=[0]*N
    sentence=[]                         #语料库,每个元素为一句不含标点的话的汉字列表
    with open(r'NLP\实验1\data\toutiao_cat_data.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
            line=line.split('_!_')      #此处line是列表(新闻ID,分类code,分类名称,新闻标题,关键词)
            line=line[3]                #此处line是字符串(只保留了标题)
            sentence.append(re.findall('[\u4e00-\u9fa5]',line))
    # print(sentence)

    for x in sentence:                  #训练A和Pi
        for i in range(len(x)):
            if i==0:                    #语料的第一个字
                if x[0] not in hanzi_list:
                    continue            #未标注的汉字直接丢弃
                else:
                    Pi[hanzi_index[x[i]]]+=1
            else:                       #x[i-1]->x[i]
                if (x[i-1] not in hanzi_list)or(x[i] not in hanzi_list):
                    continue
                else:
                    A[hanzi_index[x[i-1]]][hanzi_index[x[i]]]+=1
            print(x[i],end=" ")

    for k,v in hanzi2pinyin_dic.items():#训练B
        len_v=len(v)                    #k是汉字,v是对应拼音的集合
        for p in v:
            B[hanzi_index[k]][pinyin_index[p]]=1/len_v
```

```
    for i in range(N):                  #对A归一化
        sum=0
        for j in range(N):
            sum+=A[i][j]
        if sum==0:
            continue                    #跳过全0的状态转移
        for j in range(N):
            A[i][j]/=sum

    sum=0
    for i in range(N):                  #对Pi归一化
        sum+=Pi[i]
    for i in range(N):
        Pi[i]/=sum
    return A,B,Pi
```

3.3 Viterbi 算法

利用动态规划求解概率最大的对应观察序列的隐状态序列。算法思想见 Viterbi 搜索算法描述。

但动态规划的时间复杂度为 $O(N^2*L)$，其中 N 为语料库中汉字的数量，L 为测试语句的长度，耗时相当长。因此需要剪枝：计算$\delta_t(i)$时只分析观测状态 Ot 可能对应的隐状态，而不需要遍历所有隐状态。搜索时记录当前拼音 Ot 和前一个时刻的拼音 Ot-1，只在这两个集合间遍历即可。这样做大大降低了时间复杂度。

```
def translate(pinyin):
    '''将拼音翻译成汉字
        args: pinyin(格式为:["shen","du","xue","xi","ji","shu"])
        return: hanzi
    '''
    len_p=len(pinyin)
    delta=[[0]*N for i in range(len_p)]
    fai=[[0]*N for i in range(len_p)]
    for i in range(N):                  #初始化
        delta[0][i]=Pi[i]*B[i][pinyin_index[pinyin[0]]]
        fai[0][i]=-1

    words_pre=pinyin2hanzi_dic[pinyin[0]]       #前一个拼音可能的隐状态
    for t in range(1,len_p):            #只对可能的拼音->汉字进行查找
        words_cur=pinyin2hanzi_dic[pinyin[t]]   #当前拼音可能的隐状态
        for word_cur in words_cur:
            j=hanzi_index[word_cur]     #汉字对应的序号
            max_tmp=0
            index=-1
            for word_pre in words_pre:
                i=hanzi_index[word_pre]
                tmp=delta[t-1][i]*A[i][j]
                if tmp>=max_tmp:
                    max_tmp=tmp
                    index=i
            delta[t][j]=max_tmp*B[j][pinyin_index[pinyin[t]]]
            fai[t][j]=index
            # print("p(%s->%s)=%f"%(hanzi_list[index],word_cur,delta[t][j]),end=" ")
        words_pre=words_cur
```

```
    max_p=0                             #最终结果
    index=-1
    words=pinyin2hanzi_dic[pinyin[len_p-1]]
    for x in words:                     #只在可能的结果中搜索
        i=hanzi_index[x]
        tmp=delta[len_p-1][i]
        if tmp>=max_p:
            max_p=tmp
            index=i

    Q=[0]*len_p                         #回溯汉字
    Q[len_p-1]=index
    t=len_p-2
    while(t>=0):
        Q[t]=fai[t+1][Q[t+1]]
        t-=1
    hanzi=""
    for n in Q:
        hanzi+=hanzi_list[n]
    # print(hanzi)
    return hanzi
```

### 3.4 测试集测试模型准确性

读取测试集.txt 文件获取测试样本。从 json 文件中读取训练好的 A、B、Pi，读取测试集中每一句拼音传入维特比搜索函数获取预测结果，并将其与标签进行比较，记录正确预测的汉字与句子的比例。

需要注意的是，测试集的编码与训练集不同，前者为 GB2312，而后者和程序都是采用 UTF-8 进行编码的。直接读取出来无法进行比较，需要事先将测试集转换编码，详见问题 9。

```
with open(r'NLP\实验1\data\测试集.txt','r',encoding='UTF-8-sig') as f:
    cnt=0
    right=0
    wrong=0
    Accurate=0
    tmp=""
    while True:
        line=f.readline()
        if not line:
            break
        line=line.replace('\n','')
        if(cnt%2==0):
            line=line.casefold()            #全部转成小写
            print(line)
            line=line.split(' ')
            tmp=translate(line,hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic,A,B,Pi)
            # print(tmp)
        else:
            r,w,b=Correct(tmp,line)
            print("标签值:",line)
            print("预测值:",tmp)
            print("单句识别准确率:%f"%(r/(r+w)))
            right+=r
            wrong+=w
            Accurate+=b
        cnt+=1
print("测试集汉字预测准确率:",right/(right+wrong))
print("测试集整句预测准确率:",Accurate*2/cnt)
```

## 四、实验结果及分析和（或）源程序调试过程

**结果分析：**

通过上述过程建模求解，得到以下结果，完整代码详见文末附录。

根据结果可见，对于一些**比较常见**的句子，如“今天晚上有好看的电影”，模型可以非常**精准地识别**；但对于稍微**不常见**的句子识别效果就**不太可观**。以“四大名著”为例，头条样本集中有许多该词汇，按道理不应该识别出错。后经测试发现，头条样本集中许多词汇只存在于关键词中，而训练时只提取了标题。测试集中“钢琴”、“黑框眼镜”等词汇识别错误也是该原因导致的。

```
测试集汉字预测准确率： 0.8109452736318408
测试集整句预测准确率： 0.4782608695652174
```

总体来说，测试集识别准确率为 81.09%，可以满足对常见语句的准确识别。为了满足其他语句的识别，改进见后续章节。

测试集测试结果如下：

```
jin tian wan shang you hao kan de dian ying
标签值: 今天晚上有好看的电影
预测值: 今天晚上有好看的电影
单句识别准确率:1.000000
bei jing ao yun hui kai mu shi fei chang jing cai
标签值: 北京奥运会开幕式非常精彩
预测值: 北京奥运会开幕是非常精彩
单句识别准确率:0.916667
jin yong de wu xia xiao shuo fei chang jing cai
标签值: 金庸的武侠小说非常精彩
预测值: 金庸的武侠小说非常精彩
单句识别准确率:1.000000
shen du xue xi ji shu
标签值: 深度学习技术
预测值: 深度学习技术
单句识别准确率:1.000000
ren gong zhi neng de fa zhan
标签值: 人工智能的发展
预测值: 人工智能的发展
单句识别准确率:1.000000
zai tian an men guang chang ju xing le long zhong de yue bing shi
标签值: 在天安门广场举行了隆重的阅兵式
预测值: 在天安门广场举行了隆重的阅兵式
单句识别准确率:1.000000
qing da jia xuan ze ni jue de ke yi de shi jian
标签值: 请大家选择你觉得可以的时间
预测值: 请大家选择你觉得可以的时间
单句识别准确率:1.000000
ju you liang hao de gou tong neng li he jiao liu neng li
标签值: 具有良好的沟通能力和交流能力
预测值: 车有良好的沟通能力和交流能力
单句识别准确率:0.928571
ren zai yue du shi shi cong zuo dao you zhu zi du ru de
标签值: 人在阅读时是从左到右逐字读入的
预测值: 人在月都是失聪做到有竹子都入的
单句识别准确率:0.333333
jian shao ke cheng de ke shi shi shi fen ke xue de
标签值: 减少课程的课时是十分科学的
预测值: 减少课程的可试试试分科学的
单句识别准确率:0.692308
```

```
tou hun yan hua
标签值: 头昏眼花
预测值: 头昏眼花
单句识别准确率:1.000000
shu xin bei zhi xing ren
标签值: 失信被执行人
预测值: 舒心被执行人
单句识别准确率:0.666667
fu qi ben shi tong lin niao
标签值: 夫妻本是同林鸟
预测值: 夫妻本是同林鸟
单句识别准确率:1.000000
si da ming zhu
标签值: 四大名著
预测值: 四大明珠
单句识别准确率:0.500000
wo dai zhe yi fu hei kuang yan jing
标签值: 我戴着一副黑框眼镜
预测值: 我带着遗腹黑贶罹驳
单句识别准确率:0.333333
ta gang qin tan de bu cuo
标签值: 他钢琴弹得不错
预测值: 她港亲瘫的不错
单句识别准确率:0.285714
xiao peng you men dou xi huan qu jiao you
标签值: 小朋友们都喜欢去郊游
预测值: 小朋友们都喜欢去郊游
单句识别准确率:1.000000
预测值: 我们去三部霸
单句识别准确率:0.500000
da jia ke neng bu zhi dao
标签值: 大家可能不知道
预测值: 大家可能不知道
单句识别准确率:1.000000
wo men su she de deng huai le
标签值: 我们宿舍的灯坏了
预测值: 我们速摄的灯坏了
单句识别准确率:0.750000
测试集汉字预测准确率: 0.8109452736318408
测试集整句预测准确率: 0.4782608695652174
```

**调试过程：**

1. 问题：按行读取文件时出现“\n”；

解决方法：将读取到的整行去掉换行符：

```
line=line.replace('\n', '') #去掉结尾换行符
```

2. 问题：读取拼音时第一个拼音“a”多输出了“\ufeff”：

```
['\ufeffa', 'ai', 'an', 'ang', 'ao', 'ba', 'bai', 'ban'
'biao', 'bie', 'bin', 'bing', 'bo', 'bu', 'ca', 'cai',
```

解决方法：utf-8 编码的文件时开头会有一个多余的字符“\ufeff”，在读文件时会读到。更改编码为 UTF-8 with BOM 即可：

```
with open('NLP\实验1\data\pinyin2hanzi.txt','r',encoding='UTF-8-sig') as f:
```

3. 问题：读取 toutiao_cat_data.txt 时，语料中出现的标点无法在字典中匹配；

解决方法：使用正则表达式匹配所有汉字即可：

```
sentence.append(re.findall('[\u4e00-\u9fa5]',line))
```

4. 问题：头条的语料库中存在未被标记的汉字：

```
    A[hanzi_index[x[i-1]]][hanzi_index[x[i]]]+=1
KeyError: '陽'
```

解决方法：抛弃未标记汉字。

5. 问题：记录汉字->拼音的映射时将拼音放入字典，但是读取出来的是一个一个字符：

```
if x not in hanzi_list:
    hanzi_list.append(x)
    hanzi_index[x]=j
    j+=1
    hanzi2pinyin_dic[x]=set(line[0])
```

问题　输出　调试控制台　终端

```
{'y', 'i', 'n'}
{'a', 'y', 'yin', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'a', 'y', 'yin', 'n'}
{'y', 'i', 'n'}
{'y', 'i', 'n'}
{'q', 'yin', 'i'}
{'g', 'y', 'i', 'n'}
```

解决方法：将 `hanzi2pinyin_dic[x]=set(line[0])` 改为先创建集合再添加字符串：

```
hanzi2pinyin_dic[x]=set()
hanzi2pinyin_dic[x].add(line[0])
```

6. 问题：通过频数计算频率归一化过程中出现/0 的操作

```
    A[i][j]/=sum
ZeroDivisionError: division by zero
```

解决方法：跳过全 0 的情况

7. 问题：Viterbi 搜索算法复杂度太高，仅识别一句话的时间就接近 1 分钟：

```
深度学习技术
Execution time: 56.19025897979736 s
```

解决方法：筛选拼音可能对应的汉字，只在小集合中遍历寻找，可以看到搜索效率大大提高：

```
深度学习技术
Execution time: 0.007842063903808594 s
```

8. 问题：测试集中拼音含有大写字母，无法识别：

```
    delta[0][i]=Pi[i]*B[i][pinyin_index[pinyin[0]]]
KeyError: 'Wo'
```

解决方法：将字符串全部转换成小写：

```
line=line.casefold()                #全部转成小写
```

9. 问题：测试集编码为 GB2312，函数返回值编码为 UTF-8，无法比较；
解决方法：需要将测试集转变为 UTF-8 编码或读出后转换编码。
10. 问题：概率太小，相乘后几乎为 0，在一些测试值上无法得到正确的结果；
解决方法：将概率取对数，概率相乘改为对数相加。

```
def Log_Num(a):
    if a<=0:
        return float('-inf')
    else:
        return log(a)
```

```
for i in range(N):
    for j in range(N):
        A[i][j]=Log_Num(A[i][j])
        if(j<M):
            B[i][j]=Log_Num(B[i][j])
    Pi[i]=Log_Num(Pi[i])
```

```
delta[t][j]=max_tmp+B[j][pinyin_index[pinyin[t]]]
```

11. 问题：由于测试集一些组合的缺失导致无法识别（如下图中 hei->kuang 的转移概率是 0），最终的识别结果相当糟糕；

```
['wo', 'dai', 'zhe', 'yi', 'fu', 'hei', 'kuang', 'yan', 'jing']
标签值：我戴着一副黑框眼镜
预测值：阿阿阿阿阿阿阿阿斫
```

解决方法：在转移概率都为 0 的节点，选择任意一个继续搜索。最终回溯时只在观测值可能对应的隐状态中搜索：

```
max_p=0                                 #最终结果
index=-1
words=pinyin2hanzi_dic[pinyin[len_p-1]]
for x in words:
    i=hanzi_index[x]
    tmp=delta[len_p-1][i]
    if tmp>=max_p:
        max_p=tmp
        index=i
```

## 五、模型改进

为了解决头条样本集中许多词汇只存在于关键词中，而训练时只提取了标题导致训练结果不够乐观的情况，**将标题与关键词都加入训练集进行训练**并采取**数据平滑**。

关键词加入训练：

```
with open(r'NLP\实验1\data\toutiao_cat_data.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
        line=line.split('_!_')      #此处line是列表(新闻ID,分类code,分类名称,新闻标题,关键词)
        sentence.append(re.findall('[\u4e00-\u9fa5]',line[3]))
        if line[4]:                 #此处line[4]为关键词的字符串
            tmp=line[4].split(',')  #此处tmp为关键词的列表([普拉多,三大件,丰田,丰田普拉多,廉价版])
            for x in tmp:
                sentence.append(re.findall('[\u4e00-\u9fa5]',x))
# print(sentence)
```

拉普拉斯平滑处理数据：

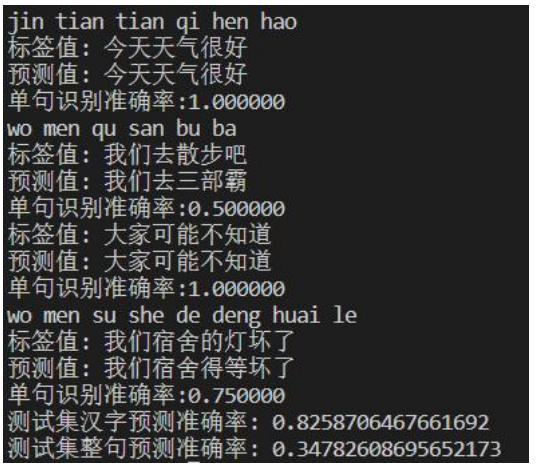
```
for i in range(N):                  #对A归一化+平滑处理
    sum=0
    for j in range(N):
        sum+=(A[i][j]+1)
    # if sum==0:
    #     continue                  #加1法不会出现全0的状态
    for j in range(N):
        A[i][j]=(A[i][j]+1)/sum

sum=0
for i in range(N):                  #对Pi归一化+平滑处理
    sum+=(Pi[i]+1)
for i in range(N):
    Pi[i]=(Pi[i]+1)/sum
```

通过上述数据处理，训练效果略有增强。

```
jin tian wan shang you hao kan de dian ying
标签值: 今天晚上有好看的电影
预测值: 今天晚上有好看的电影
单句识别准确率:1.000000
bei jing ao yun hui kai mu shi fei chang jing cai
标签值: 北京奥运会开幕式非常精彩
预测值: 北京奥运会开幕式非常精彩
单句识别准确率:1.000000
jin yong de wu xia xiao shuo fei chang jing cai
标签值: 金庸的武侠小说非常精彩
预测值: 金用的武侠小说非常精彩
单句识别准确率:0.909091
shen du xue xi ji shu
标签值: 深度学习技术
预测值: 深度学习技术
单句识别准确率:1.000000
```

```
ren gong zhi neng de fa zhan
标签值: 人工智能的发展
预测值: 人工智能的发展
单句识别准确率:1.000000
zai tian an men guang chang ju xing le long zhong de yue bing shi
标签值: 在天安门广场举行了隆重的阅兵式
预测值: 在天安门广场举行了隆中的阅兵式
单句识别准确率:0.933333
qing da jia xuan ze ni jue de ke yi de shi jian
标签值: 请大家选择你觉得可以的时间
预测值: 清大家选择你觉得可以的时间
单句识别准确率:0.923077
ju you liang hao de gou tong neng li he jiao liu neng li
标签值: 具有良好的沟通能力和交流能力
预测值: 车有量好的沟通能力和交流能力
单句识别准确率:0.857143
ren zai yue du shi shi cong zuo dao you zhu zi du ru de
标签值: 人在阅读时是从左到右逐字读入的
预测值: 人在月都是什聪做到有竹子都入的
单句识别准确率:0.333333
jian shao ke cheng de ke shi shi shi fen ke xue de
标签值: 减少课程的课时是十分科学的
预测值: 减少课程的可是世十分科学的
单句识别准确率:0.769231
tou hun yan hua
标签值: 头昏眼花
预测值: 投昏眼花
单句识别准确率:0.750000
shu xin bei zhi xing ren
标签值: 失信被执行人
预测值: 书信被执行人
单句识别准确率:0.833333
fu qi ben shi tong lin niao
标签值: 夫妻本是同林鸟
预测值: 夫妻本是同林鸟
单句识别准确率:1.000000
si da ming zhu
标签值: 四大名著
预测值: 四大明珠
单句识别准确率:0.500000
```

```
wo dai zhe yi fu hei kuang yan jing
标签值: 我戴着一副黑框眼镜
预测值: 我带着一夫黑框眼睛
单句识别准确率:0.666667
ta gang qin tan de bu cuo
标签值: 他钢琴弹得不错
预测值: 塔钢琴弹的不错
单句识别准确率:0.714286
xiao peng you men dou xi huan qu jiao you
标签值: 小朋友们都喜欢去郊游
预测值: 小朋友们都喜欢去较有
单句识别准确率:0.800000
zhe ge wan ju hen you qu
标签值: 这个玩具很有趣
预测值: 这个玩具很有趣
单句识别准确率:1.000000
ta hen sheng qi
标签值: 他很生气
预测值: 她很生气
单句识别准确率:0.750000
```




其实不能只观察汉字识别正确率，对比改进前后，改进后二元词组的识别效率明显增强，这归功于 HMM 模型采用的是二元文法。改进后还有一些句中二元**词组组合正确但不适合语境**的情况，例如“他”和“她”，“散步”和“三部”，“眼睛”和“眼镜”。

## 下面提出一些**改进意见**：

1. 增强数据集的词汇量以及**词汇的使用场景**，这对语境下使用正确的词很有帮助；
2. 增加**三元、四元文法**训练可以增强训练集的泛化能力；
3. 增加训练集的**语言模式**：如本实验中测试集中大多为**口语祈使句**或**感叹句**，只有少量的文学语法；但训练集取自新闻标题，多为简洁的**文学**语法，这样训练下来的模型对包含较多口语的训练集效果不够乐观；
4. 增加训练集的**语言结构**：本实验中多为词组和短句，可以增加整段甚至整篇文章进行训练，也许会提高识别正确率。

以上只是笔者的浅薄只见，如何改进还需由老师和助教们定夺。

## 六、附录（完整代码）

DataProcessing.py

```
'''预处理训练语料并训练 HMM 模型
数据预处理:
    读取 pinyin2hanzi.txt 文件,得到所有汉字、拼音各自的列表,并记录其编号(显隐状态都由编号指代)
    为了处理一字多音和一音多字,记录了同一个字对应的不同读音和同一个读音对应的不同字
训练 HMM 模型:
    读取 toutiao_cat_data.txt 文件,训练得到状态转移概率矩阵 A 和初始状态矩阵π
    用预处理得到的汉字——>多个读音取均值近似 Pi
'''
import re
import time
import json

def Construct_Dict():
    '''读取 pinyin2hanzi.txt 文件,构造隐显状态集合,即汉字和拼音的列表并记录其下标
        return:
            hanzi_list:隐状态的列表
            pinyin_list:显状态的列表
            hanzi_index:隐状态对应的 hanzi_list 中坐标的字典
            pinyin_index:显状态对应的 pinyin_list 中坐标的字典
            hanzi2pinyin_dic:隐状态可能对应的显状态集合的字典
            pinyin_dic:显状态可能对应的隐状态集合的字典
    '''
    hanzi_list=[]
    pinyin_list=[]
    hanzi_index={}
    pinyin_index={}
    hanzi2pinyin_dic={}
    pinyin2hanzi_dic={}
    i=0
    j=0
    with open(r'NLP\实验 1\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
            pinyin_list.append(line[0])
            pinyin_index[line[0]]=i
            i+=1
            pinyin2hanzi_dic[line[0]]=set()
            for x in line[1]:
                pinyin2hanzi_dic[line[0]].add(x)
                if x not in hanzi_list: #x 是第一次出现的汉字
                    hanzi_list.append(x)
                    hanzi_index[x]=j
                    j+=1
                    # hanzi2pinyin_dic[x]=set(line[0])  #字符串 line[0]会被拆成多个字符
                    hanzi2pinyin_dic[x]=set()
                    hanzi2pinyin_dic[x].add(line[0])
                else:                        #x 是多音字
                    hanzi2pinyin_dic[x].add(line[0])
    return hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic

def
Train_miu(hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic
):
    '''读取 toutiao_cat_data.txt 文件,构造状态转移概率矩阵 A,发射概率矩阵 B,初始状态
概率矩阵 Pi
        return: A,B,Pi
    '''
    N=len(hanzi_list)
    M=len(pinyin_list)
    A=[[0]*N for i in range(N)]
    B=[[0]*M for i in range(N)]
    Pi=[0]*N
    sentence=[]                          #语料库,每个元素为一句不含标点的话的汉字列表
    with open(r'NLP\实验 1\data\toutiao_cat_data.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
```

```
            line=line.split('_!_')      #此处 line 是列表(新闻 ID,分类 code,分类名称,新闻标题,关键词)
            line=line[3]                #此处 line 是字符串(只保留了标题)
            sentence.append(re.findall('[\u4e00-\u9fa5]',line))
    # print(sentence)

    for x in sentence:                  #训练 A 和 Pi
        for i in range(len(x)):
            if i==0:                    #语料的第一个字
                if x[0] not in hanzi_list:
                    continue            #未标注的汉字直接丢弃
                else:
                    Pi[hanzi_index[x[i]]]+=1
            else:                       #x[i-1]->x[i]
                if (x[i-1] not in hanzi_list)or(x[i] not in hanzi_list):
                    continue
                else:
                    A[hanzi_index[x[i-1]]][hanzi_index[x[i]]]+=1
            print(x[i],end=" ")

    for k,v in hanzi2pinyin_dic.items():#训练 B
        len_v=len(v)                    #k 是汉字,v 是对应拼音的集合
        for p in v:
            B[hanzi_index[k]][pinyin_index[p]]=1/len_v

    for i in range(N):                  #对 A 归一化
        sum=0
        for j in range(N):
            sum+=A[i][j]
        if sum==0:
            continue                    #跳过全 0 的状态转移
        for j in range(N):
            A[i][j]/=sum

    sum=0
    for i in range(N):                  #对 Pi 归一化
        sum+=Pi[i]
    for i in range(N):
        Pi[i]/=sum
```

```
    return A,B,Pi

# time_start = time.time()
#
hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic=Construct
_Dict()
#
A,B,Pi=Train_miu(hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2ha
nzi_dic)
# time_end = time.time()
# time_c= time_end - time_start
# print('\nExecution time:', time_c, 's')

# with open(r"NLP\实验 1\train_data\A.json","w") as f:
#       json.dump(A,f)
# with open(r"NLP\实验 1\train_data\B.json","w") as f:
#       json.dump(B,f)
# with open(r"NLP\实验 1\train_data\Pi.json","w") as f:
#       json.dump(Pi,f)
# print("Writing finished!")
```

Viterbi.py

```
'''Viterbi 搜索算法
    delta[t][i]表示时间 t 时状态为 Si 且观测序列为 O1O2...Ot 的最大概率
    fai[t][j]表示 t 时的状态 Si 的前驱节点
初始状态:
    delta[1][i]=Pi[i]*B[i][O1]
    fai[1][j]=-1
状态转移:
    delta[t][j]=max{delta[t-1][i]*A[i][j]}*B[j][Ot]
    fai[t][j]=argmax(i){delta[t-1][i]*A[i][j]}*B[j][Ot]
最终结果:
    QT=argmax{delta[T][i]}
回溯:
    q[t]=fai[t+1][q[t+1]]
'''
import json
import time
from DataProcessing import Construct_Dict
```

```
# with open(r"NLP\实验 1\train_data\A.json") as f:
#       A=json.load(f)
# with open(r"NLP\实验 1\train_data\B.json") as f:
#       B=json.load(f)
# with open(r"NLP\实验 1\train_data\Pi.json") as f:
#       Pi=json.load(f)
#
hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic=Construct
_Dict()

def
translate(pinyin,hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2han
zi_dic,A,B,Pi):
    '''将拼音翻译成汉字
        args: pinyin(格式为:["shen","du","xue","xi","ji","shu"])
            hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic
        return: hanzi
    '''
    N=len(A)
    M=len(B[0])
    len_p=len(pinyin)
    delta=[[0]*N for i in range(len_p)]
    fai=[[0]*N for i in range(len_p)]
    for i in range(N):                      #初始化
        delta[0][i]=Pi[i]*B[i][pinyin_index[pinyin[0]]]
        fai[0][i]=-1

    # for t in range(1,len_p):              #动态规划(复杂度太高)
    #       for j in range(N):
    #           max_tmp=0
    #           index=-1
    #           for i in range(N):
    #               tmp=delta[t-1][i]*A[i][j]
    #               if tmp>max_tmp:
    #                   max_tmp=tmp
    #                   index=i
    #           delta[t][j]=max_tmp*B[j][pinyin_index[pinyin[t]]]
    #           fai[t][j]=index
```

```
    words_pre=pinyin2hanzi_dic[pinyin[0]]        #前一个拼音可能的隐状态
    for t in range(1,len_p):                #只对可能的拼音->汉字进行查找
        words_cur=pinyin2hanzi_dic[pinyin[t]]    #当前拼音可能的隐状态
        for word_cur in words_cur:
            j=hanzi_index[word_cur]       #汉字对应的序号
            max_tmp=0
            index=-1
            for word_pre in words_pre:
                i=hanzi_index[word_pre]
                tmp=delta[t-1][i]*A[i][j]
                if tmp>=max_tmp:
                    max_tmp=tmp
                    index=i
            delta[t][j]=max_tmp*B[j][pinyin_index[pinyin[t]]]
            fai[t][j]=index
            # print("p(%s->%s)=%f"%(hanzi_list[index],word_cur,delta[t][j]),end=" ")
        words_pre=words_cur

    max_p=0                                 #最终结果
    index=-1
    words=pinyin2hanzi_dic[pinyin[len_p-1]]
    for x in words:                         #只在可能的结果中搜索
        i=hanzi_index[x]
        tmp=delta[len_p-1][i]
        if tmp>=max_p:
            max_p=tmp
            index=i

    Q=[0]*len_p                             #回溯汉字
    Q[len_p-1]=index
    t=len_p-2
    while(t>=0):
        Q[t]=fai[t+1][Q[t+1]]
        t-=1
    hanzi=""
    for n in Q:
        hanzi+=hanzi_list[n]
    # print(hanzi)
    return hanzi
```

```
# time_start = time.time()
#
print(translate(["shen","du","xue","xi","ji","shu"],hanzi_list,pinyin_list,hanzi_index,pinyin_in
dex,hanzi2pinyin_dic,pinyin2hanzi_dic,A,B,Pi))
# time_end = time.time()
# time_c= time_end - time_start
# print('Execution time:', time_c, 's')
```

main.py

```
'''基于 HMM 的拼音转汉字程序
    预处理训练语料
    训练 HMM 模型
    维特比算法实现从任意拼音到汉字的自动转换
    测试集评价程序的转换准确率
'''
import json
from DataProcessing import Construct_Dict
from Viterbi import translate

def Correct(s1,s2):                    #比较正确/错误字符数以及字符串是否相同
    n=len(s1)
    right=0
    wrong=0
    for i in range(n):
        if(s1[i]==s2[i]):
            right+=1
        else:
            wrong+=1
    return right,wrong,wrong==0

with open(r"NLP\实验 1\train_data\A.json") as f:
    A=json.load(f)
with open(r"NLP\实验 1\train_data\B.json") as f:
    B=json.load(f)
with open(r"NLP\实验 1\train_data\Pi.json") as f:
    Pi=json.load(f)

hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_dic,pinyin2hanzi_dic=Construct
```

```
_Dict()
#
print(translate(["shen","du","xue","xi","ji","shu"],hanzi_list,pinyin_list,hanzi_index,pinyin_in
dex,hanzi2pinyin_dic,pinyin2hanzi_dic,A,B,Pi))

with open(r'NLP\实验 1\data\测试集.txt','r',encoding='UTF-8-sig') as f:
    cnt=0
    right=0
    wrong=0
    Accurate=0
    tmp=""
    while True:
        line=f.readline()
        if not line:
            break
        line=line.replace('\n','')
        if(cnt%2==0):
            line=line.casefold()              #全部转成小写
            print(line)
            line=line.split(' ')
            tmp=translate(line,hanzi_list,pinyin_list,hanzi_index,pinyin_index,hanzi2pinyin_d
ic,pinyin2hanzi_dic,A,B,Pi)
            # print(tmp)
        else:
            r,w,b=Correct(tmp,line)
            print("标签值:",line)
            print("预测值:",tmp)
            print("单句识别准确率:%f"%(r/(r+w)))
            right+=r
            wrong+=w
            Accurate+=b
        cnt+=1
print("测试集汉字预测准确率:",right/(right+wrong))
print("测试集整句预测准确率:",Accurate*2/cnt)
```