学习材料加工及其代码

5.3 学习材料

本节主要介绍自适应词汇学习系统所使用的学习材料，包括词汇筛选、词汇例句的选择、词汇义项的选择、词汇搭配的选择、词汇其他学习内容等。教学实验中实验组与对照组均使用本节所阐述的学习材料。

5.3.1 词汇筛选

国家汉办于 2009 年推出了新汉语水平考试，并于 2012 年修订了新汉语水平考试词汇（2012 年修订版）[1]。本系统所研究的对外汉语词汇就是以新汉语水平考试词汇（2012 年修订版）为标准。中级对外汉语词汇包括新汉语水平考试词汇（2012 年修订版）的新 HSK 三级词汇（共 300 个词汇）和新 HSK 四级词汇（共 600 个词汇）。为了方便研究，本文以新 HSK 四级词汇为筛选的前提。

词频作为词汇筛选的关键因素。本研究结合大型语料库和教材语料库对新 HSK 四级词汇进行词频统计。语料库选择北京语言大学 BCC 语料库[2][78]，本研究选用了 BCC 语料库汉语词频表的 Global 版本[3]。笔者使用 Pandas、SQL，以 BCC 语料库汉语词频表的 Global 版本为依据，对新 HSK 四级词汇的词频进行了对照查询，并将新 HSK 四级词汇的词频文件保存到本地，以便后续进行进一步处理。程序见附录 A。关于教材语料，笔者选取了主流的中级对外汉语教材，包括《博雅汉语》、《成功之路》、《发展汉语》、《HSK 标准教程》四个版本，共计 13 本[4]。教材语料的处理步骤如下：（1）使用 OCR 文字识别软件 ABBYY FineReader 12 将 PDF 版本的教材（13 个 PDF 文件）转化为 word 版本（13 个 word 文件），将 word 版本中的课文语料，复制到 txt 文件（13 个 txt 文件）中。（2）使用 Python 与正则表达式，匹配 txt 格式的课文语料中的中文与中文标点符号，并且去除空格，保存为新版本的 txt 课文语料文档（13 个清洗过的的 txt 文件）。程序见附录 B。（3）将 13 个清洗过的 txt 文件合并为一个新的 txt 文件，也就是经过清洗的 13 本中级对外汉语教材的课文语料集合（以下简称最终版本课文语料）。程

---

[1] http://www.chinesetest.cn/godownload.do

[2] http://bcc.blcu.edu.cn/

[3] BCC 语料库汉语部分分为 news、technology、blog、weibo 和 literature 几个频道。Global 为五个频道合并后的全局词表。下载链接为 http://bcc.blcu.edu.cn/的下载-词汇资源-BCC 汉语词频表。

[4] 具体为《博雅汉语中级冲刺篇 1》、《博雅汉语中级冲刺篇 2》、《博雅汉语中级加速篇 1》、《博雅汉语中级加速篇 2》;《成功之路跨越篇 1》、《成功之路跨越篇 2》、《成功之路提高篇 1》、《成功之路提高篇 2》;《发展汉语中级综合 1》、《发展汉语中级综合 2》;《HSK 标准教程 3》、《HSK 标准教程 4（上）》、《HSK 标准教程 4（下）》。

序见附录 C。经过统计，新的 txt 文件字数为 16.5 万字。（4）对最终版本课文语料进行所有课文词汇的词频统计。首先使用结巴分词对课文语料进行分词，然后使用 python 进行所有词汇的词频统计，并保存为 DataFrame 格式，按照词频进行排序后，将此 DataFrame 格式文件保存为教材语料词频表。程序文件见附录 D。（5）筛选教材语料词频表中的新 HSK 四级词汇及词频，并与基于 BCC 语料库的新 HSK 四级词汇及词频表进行合并，合并为新的四级词汇词频表 new_grade4_word_freq。程序文件见附录 E。（6）对新的四级词汇词频表 new_grade4_word_freq 进行数据清洗、特征选择、数据规范化。由于在 600 个新 HSK 四级词汇中，有 10 个词汇不能在 BCC 语料库词频表中查询到，为了后续进行词频的相加计算，将 10 个词汇的空值词频填充为 0。选择 new_grade4_word_freq 中的 vocabulary 列、frequency 列、textbook_freq 列。由于基于 BCC 语料库查询的新 HSK 四级词汇词频与基于教材语料库查询的新 HSK 四级词汇词频的量级存在较大差异，因此，对 new_grade4_word_freq 中的 frequency 列、textbook_freq 列进行最小-最大规范化。程序文件见附录 F。（7）教材语料是专门面向学习者的内容，其词汇更有针对性。BCC 语料库涉及更多广范围的语料，其词汇词频可以作为参考。本研究将新 HSK 四级词汇所对应的教材语料词频的权重设为 0.7，将所对应的 BCC 语料库词频的权重设为 0.3，并计算总的频率。程序见文件同样见附录 F。

在基于 BCC 语料库和教材语料库完成新 HSK 四级词汇的词频统计，并按词频进行排序之后，首先以高频、中频与低频三类词频进行词汇选取。进行如此选择，是因为在先导试验中，学习者对于高频词掌握程度较好，因此不宜只是学习高频词。除了以词频为依据进行词汇的选择外，还会根据词性进行选择。由于虚词没有实在的词汇意义，更多反映了语法知识，而不是词语知识[137]。本研究在选取词汇时，主要选择实词，并以名词、动词、形容词为主。

除了按照词频、词性进行词汇的选取之外，本文还研究了易混淆词的学习。因此，还会着重选取一些包含易混淆词的词汇，并且易混淆词为新 HSK 一级到三级词汇。

易混淆词的处理。易混淆词的界定标准参考胡韧奋[33]44 关于词语偏误信息的研究：目标词偏误频次在十次以上，且目标词误用为该词的频次占偏误总频次的 20%以上。易混淆词例句来自于北京语言大学动态作文语料库[5]。易混淆词提取程序见附录 G。详细操作步骤如下：

（1）从北京语言大学动态作文语料库下载使用了目标词的句子；

[5] http://hsk.blcu.edu.cn/

（2）筛选使用出错的目标词；

（3）提取目标词的混淆词；

（4）查询每个混淆词出现的次数并排序；

（5）确定是否为易混淆词，并筛选易混淆词；

（6）查询易混淆词等级，筛选为新 HSK 一级到三级词汇的易混淆词；

（7）把目标词的易混淆词写入到 csv 文件。

综上所述，本研究在筛选词汇时，会基于 BCC 语料库和教材语料库进行词频统计，筛选高频词、中频词、低频词，重点选择名词、动词、形容词，并着重选择一些包含易混淆词的词汇。筛选后的词汇见附录 W。

筛选后的词汇分为六次进行学习。每次学习安排相当高频词、中频词、低频词的学习，易混淆词的学习安排在目标词学习之后。六次学习在此分别称作 part1、part2、part3、part4、part5、part6。详见附录 X。

5.3.2 词汇例句的选择

在选择好要学习的词汇之后，对词汇进行例句的选择。首先筛选教材语料库中包含特定词汇的句子。然后进行句子难度的排序。最后基于句子难度以及人工确认，来确认例句。具体如下：

（1）例句选择来源于清洗过后的教材语料库。本研究使用 re 模块、Pandas、结巴分词对教材语料库进行词汇例句的筛选。程序见附录 H。

（2）使用结巴分词对句子进行分词，使用 Pandas 添加分词列。为了判断句子中除了目标词汇是新 HSK 四级词汇，其他均为新 HSK 一级词汇、二级词汇、三级词汇，本文使用 Pandas 将目标词汇也添加到分词中，并设置为新分词列，然后使用程序进行判断。经过统计发现，在教材语料中，对于 60 多个目标词汇来说，只有 2 个句子是仅包含目标词汇和新 HSK 一级、二级、三级词汇的。这不能满足实验学习的要求。程序见附录 I。为了进一步验证，笔者进一步使用 HSK 常用词例句文件进行搜索，发现没有任何一个句子仅包含目标词汇和新 HSK 一级、二级、三级词汇的。程序见附录 J。因此，例句筛选时，不再局限于只包含目标词汇和新 HSK 一级、二级、三级词汇，而是尽量筛选除了待学习的四级目标词汇外，尽量少包含新 HSK 四级、五级、六级词汇的句子。

（3）统计每个每个句子出现的高等级词汇[6]的个数，并列出句子中出现的高等级词汇。程序见附录 K。

（4）统计每个句子的难度。胡韧奋认为影响句子难度的因素包括句子包含的语法、句长、词语等级信息[33]45。本文在研究句子难度时，不对句子包含的语

[6] 此处指新 HSK 四级、五级、六级词汇。

法进行相关研究，将句子难度聚焦于句长与词语等级信息两个方面。这两个方面的计算可以将句子所包含的词汇难度相加，因为词汇难度即可在相当程度上反映词汇等级信息，词汇数量相加则可以反映出来句长。关于词汇难度，J. Stewart 等[17]研究词汇难度与词频的关系，得出词汇难度与词频的对数呈现强相关，相关性可以达到 0.8。基于这一结论，本文首先进行词汇难度的计算，笔者选用的是 BCC 语料库词频表，对词频取对数后再取倒数，然后进行最小-最大规范化，将词汇难度变换到[0,1]的空间中。将词汇难度添加到词频表中，进行特征选取，只留下词频及词汇难度，保存为 csv 文件。将 csv 文件保存为词汇复杂度字典。程序见附录 L。句子复杂度计算则将句子中每个词汇的难度相加，程序见附录 M，主要步骤为：（a）读取文件；（b）根据词汇复杂度计算句子复杂度；（c）为方便查看句子复杂度，对句子复杂度进行最小-最大标准化；（d）将标准化后的句子复杂度添加到文件；（e）保存句子复杂度文件。

（5）对于每个词汇，在对句子按照句子复杂度升序排列后，人工查看排名前 20 左右的句子，统计其词汇义项，筛选出现频率较高的义项。对于每个出现频率较高的词汇义项，挑选一个句子。挑选句子以句子复杂度为标准，挑选句子复杂度低的句子；句子复杂度相当时，挑选出现高等级词汇少的句子。

5.3.3 词汇义项的选择

词汇义项的选择分为两个步骤：

（1）查询 PIeco 词典[7]。人工确定可能的词汇高频义项。

（2）对于每个词汇，将句子复杂度由低到高排列后，人工查看排名前 20 左右的句子，统计其词汇义项，筛选出现频率较高的义项。

5.3.4 词汇搭配的选择

胡韧奋等[26]135-144使用语言特征方法、自然语言处理技术、统计方法对语料库进行搭配的自动抽取，在对外汉语教材语料库（约 240 万词）和中文维基百科语料库（约 1.38 亿词）中分别抽取了 22298 条搭配与 219451 条搭配，构建了两个大规模搭配知识库；同时基于搭配知识库构建了中文搭配助手[8]，服务对外汉语教学与信息处理应用。

为了获取想要的词汇搭配，笔者首先使用中文搭配助手进行词汇搭配查询，将查询所得的内容保存到 Excel 文件中，然后用 python 进行对 Excel 文件进行内容方面的处理，提取搭配，标注搭配词汇的词汇等级。根据 i+1 的原则，要学

---

[7] Pleco 词典内置两款免费词典：一是流行的开源 CC- CEDICT，二是 Pleco 自己的 PLC 词典。官方网站：http://www.pieco.in/

[8] 在线中文搭配检索平台，网址为 http://cca.xingtanlu.cn/

习的搭配应该是学习者之前学习的词汇，以建立词汇之间的连接，因此，笔者对搭配词汇进行筛选，筛选出新 HSK 一级、二级、三级词汇，并按照互信息由高到低降序排列。程序见附录 N。详细操作步骤如下：

（1）读取词汇搭配 excel 文件；

（2）使用正则表达式提取出搭配词汇，去除目标词和空格；

（3）根据新 HSK 词汇表，识别搭配词汇中的新 HSK 词汇以及词汇等级，去除搭配词汇中的非新 HSK 词汇；

（4）筛选搭配词汇中新 HSK 一级、二级、三级词汇，并按照互信息进行降序排列；

（5）保存为 csv 文件。

对于每个词汇的每个义项，人工选择两个互信息高的搭配词汇进行学习。

5.3.5 词汇其他学习内容

1、字。词由字所组成。结合词汇来确定字的词性、英文释义，参考自 PIeco 词典。字的拼音来自百度汉语[9]。字的发音文件来自百度翻译[10]。

2、词汇。词汇词性、英文释义参考自 PIeco 词典。词的拼音来自百度汉语[9]。词的发音文件来自百度翻译[10]。使用 requests 库爬取字、词、搭配、句子发音，程序见附录 O。由于爬取的发音文件为 mp3 格式，将 mp3 格式的音频文件插入到 ppt 中时，并不是内嵌到 ppt 中的，交给学习者后，学习者使用时不太方便，因此，将 mp3 格式的音频文件转换成 wav 格式[11]。程序文件见附录 P。

3、搭配。搭配的英文释义来自谷歌翻译[12]，并进行人工确认或修改。搭配的发音处理同词汇相同。

4、句子。句子分词与词汇拼音标注使用语料库在线的汉语拼音自动标注功能[13]。句子的英文释义来自谷歌翻译[12]，并进行人工确认或修改。句子的发音处理同词汇相同。

5、词汇释义练习题、词汇搭配练习题选项的制定。词汇释义练习题、词汇搭配练习题的选项均由 A、B、C、D 四个选项组成，除了其中一个选项为正确选项外，其他选项从 HSK 一级到 HSK 三级词汇中随机选择。使用 random 模块进行词汇的随机选择，程序见附录 AD。对于词汇搭配练习题，如果随机选择的选项除了要学习的词汇作为正确选项外，还出现了其他正确的选项，那么就将其他正确

---

[9] https://hanyu.baidu.com/

[10] https://fanyi.baidu.com/

[11] Wav 格式的音频文件插入到 ppt 中时，是内嵌到 ppt 中的，方便传递给学习者。

[12] https://translate.google.cn/

[13] http://corpus.zhonghuayuwen.org/CpsPinyinTagger.aspx

的选项换掉，也就是再次从 HSK 一级到 HSK 三级词汇中随机选择词汇，直到词汇选项完全符合要求，也就是除了要学习的词汇作为正确选项外，其他选项均不能作为正确选项。

附录 A 基于 BCC 语料库词频表查询新 HSK 四级词汇词频

```
# 获取词表中的四级词汇
import pandas as pd
df = pd.read_excel('/Users/kangzhengwei/Desktop/HSK-2012.xls', header=None)
df4 = df[df.iloc[:, 0].str.contains('四级')]
df4['等级'] = df4.iloc[:, 0].str[-3:-1]
df4['词汇'] = df4.iloc[:, 0].str[:-4]
df4

# 读取 BCC 语料库词频
import pandas as pd

word_freq = pd.read_csv('./global_wordfreq.txt', sep='\t', header=None)
word_freq

# 修改 df4 列名
df4 = df4.rename(columns={0:'original','等级':'grade', '词汇':'vocabulary'})
df4

# 修改 word_freq 列名
word_freq= word_freq.rename(columns={0:'vocabulary',1:'frequency'})
word_freq

# 建立 四级词汇表 与 词频表 的左连接

import pandas as pd
from pandas import DataFrame
from pandasql import sqldf
```

```
    pysqldf = lambda sql: sqldf(sql, globals())
    sql = "select df4.vocabulary, word_freq.frequency from df4 left join word_freq on df4.vocabulary == word_freq.vocabulary"

    grade4_word_freq = pysqldf(sql)
    grade4_word_freq

    # 对四级词汇按照词频进行降序排列
    grade4_word_freq = grade4_word_freq.sort_values(by='frequency', ascending=False)
    grade4_word_freq

    # 写入到 excel 文件中
    grade4_word_freq.to_excel('/Users/kangzhengwei/Desktop/新实验/四级词汇即及词频.xlsx')
```

附录 B Python 正则表达式匹配课文语料中的中文与中文标点

```
import re

for info in os.listdir('/Users/kangzhengwei/Desktop/课文语料/课文语料 txt'):
        domain = os.path.abspath('/Users/kangzhengwei/Desktop/课文语料/课文语料 txt')
        complete_path = os.path.join(domain, info)  #将路径与文件名结合起来就是每个文件的完整路径
        print(complete_path)
        information = open(complete_path, 'r')
        a = information.read()

        t1 = re.findall('[\u3002\uff1b\uff0c\uff1a\u201c\u201d\uff08\uff09\u3001\uff1f\u300a\u300b\u4e00-\u9fa5]',a)
        d = ''.join(t1)

        with open(complete_path[:-4] + 'new.txt', 'w') as f:
            f.write(d)
```

附录 C 合并教材语料

```
# 字符串添加
total_corpus = ''

import re

for info in os.listdir('/Users/kangzhengwei/Desktop/课文语料/无空格的语料'):
    domain = os.path.abspath('/Users/kangzhengwei/Desktop/课文语料/无空格的语料')
    complete_path = os.path.join(domain, info)  #将路径与文件名结合起来就是每个文件的完整路径
    print(complete_path)
    information = open(complete_path, 'r')
    a = information.read()

    total_corpus += a


with open('/Users/kangzhengwei/Desktop/课文语料/所有的中级教材语料', 'w') as f:
        f.write(total_corpus)
```

附录 D 统计教材语料词频

```
import jieba
import pandas as pd
from pandas import DataFrame

txt = open('/Users/kangzhengwei/Desktop/课文语料/所有的中级教材语料.txt', 'r').read()
words = jieba.lcut(txt)  #分词 直接返回 list

counts = {}  #词频统计

# 要查询的词汇
df = pd.read_excel('/Users/kangzhengwei/Desktop/新实验/四级词汇即及词频.xlsx')
search_words = df['vocabulary'].values

for word in words:
    counts[word] = counts.get(word, 0) + 1
for key, value in counts.items():
    print(key, value)


# 把词频统计数据写入到 DataFrame 中

textbook_words_freq = pd.DataFrame(columns=['textbook_word','textbook_freq'])

for key, value in counts.items():
    new = pd.DataFrame({'textbook_word': key,'textbook_freq': value
    }, index=[0])
    textbook_words_freq = textbook_words_freq.append(new, ignore_index=True)
```

```
    textbook_words_freq

    textbook_words_freq = textbook_words_freq.sort_values(by='textbook_freq',ascending=False)
    textbook_words_freq

    textbook_words_freq.to_excel('/Users/kangzhengwei/Desktop/新实验/教材语料词频表.xlsx')
```

附录 E 筛选教材词汇中的新 HSK 四级词汇及词频，并合并基于 BCC 语料库的词汇及词频统计表

```
# 读取基于 BCC 语料库的四级词汇及词频表

grade4_words_freq = pd.read_excel(
    "/Users/kangzhengwei/Desktop/新实验/四级词汇即及词频.xlsx")
grade4_words_freq

# 筛选教材词汇中的新 HSK 四级词汇及词频，并合并基于 BCC 语料库的词汇及词频统计表

import pandas as pd
from pandas import DataFrame
from pandasql import sqldf

pysqldf = lambda sql: sqldf(sql, globals())
sql = "select grade4_words_freq.vocabulary, grade4_words_freq.frequency, textbook_words_freq.textbook_word, textbook_words_freq.textbook_freq from grade4_words_freq left join textbook_words_freq on grade4_words_freq.vocabulary == textbook_words_freq.textbook_word"

new_grade4_word_freq = pysqldf(sql)
new_grade4_word_freq

new_grade4_word_freq.to_excel('/Users/kangzhengwei/Desktop/新实验/new_grade4_word_freq.xlsx')
new_grade4_word_freq
```

附录 F 对 new_grade4_word_freq 进行数据清洗、特征选择、数据规范化

```
    # 填充空值
    new_grade4_word_freq.fillna(0, inplace=True)
    new_grade4_word_freq

    # 对 new_grade4_word_freq 进行特征选择及数据标准化

    # 特征选择

    new_grade4_word_freq = new_grade4_word_freq[[
        'vocabulary', 'frequency', 'textbook_freq'
    ]]
    new_grade4_word_freq

    from sklearn import preprocessing
    import numpy as np
    x = new_grade4_word_freq[['frequency', 'textbook_freq']]
    # 将数据进行[0,1]规范化
    min_max_scaler = preprocessing.MinMaxScaler()
    minmax_x = min_max_scaler.fit_transform(x)
    print(minmax_x)

    new_grade4_word_freq[['BCC_freq_MinMax','text_freq_MinMax']]    =
minmax_x
    new_grade4_word_freq

    new_grade4_word_freq['last_freq'] = 0.3 * new_grade4_word_freq[
        'BCC_freq_MinMax']             +             0.7             *
new_grade4_word_freq['text_freq_MinMax']
    new_grade4_word_freq
```

```
    last_grade4_word_freq =
last_grade4_word_freq.sort_values(by='last_freq', ascending=False)
    last_grade4_word_freq

    last_grade4_word_freq.to_excel(
        '/Users/kangzhengwei/Desktop/新实验
/last_grade4_word_freq.xlsx')
```

附录 G 易混淆词提取

```
    # 读取数据
    select_word = '原因'
    #显示所有行
    # pd.set_option('display.max_rows', None)
    #设置 value 的显示长度为 100，默认为 50
    pd.set_option('max_colwidth', 100)

    import pandas as pd
    from pandas import DataFrame

    # https://blog.csdn.net/qq_32650831/article/details/121000332
    data = pd.read_csv('/Users/kangzhengwei/Desktop/新实验/易混淆词/'
+ select_word +
                       '.csv',
                       index_col=False)
    data

    # 筛选出错的目标词
    df1 = data[data['检索原句'].str.contains(select_word+'{CC') ==
True]
    df1

    # 提取混淆词
    # 第一步 提取 目标词 + 后面的易混淆词 + 标注
    df2 = df1['检索原句'].str.extract(r'(' + select_word +
                                    '\{CC\d*[\u4e00-\u9fa5]+\[?[A-
Z]*\]?\})',
                                    expand=False)
    df2

    # 正确的第二步 删除目标词
    df3 = df2.str.extract(r'([^(' + select_word + ')].+)',
```

```
expand=False)
    df3

    # 第三步：提取易混淆词
    df4 = df3.str.extract(r'([\u4e00-\u9fa5]+)')
    df4

    # 查询每个混淆词出现的次数并排序
    df5 = df4.groupby(df4.iloc[:,0]).count()
    df5

    # 确定是否为易混淆词并筛选易混淆词
    import numpy as np
    # 计算错误总频次
    error_total_time = np.sum(df5.iloc[:, 0].values, axis=0)
    # 计算行数
    total_rows = df5.shape[0]
    # 设置错误总频次数组
    df_sum = np.array([error_total_time] * total_rows)
    # 添加错误总频次列
    # df5 的 index 也得加进来，否则 concat 不是想要的结果
    df_add_errorTotalTime = pd.concat(
        [df5, pd.DataFrame(df_sum, index=df5.index)], axis=1)
    # 添加 目标词误用为该词的频次占偏误总频次的占比
    # 可以这样除啊
    df_add_errorTotalTime[
        'error_rate'] = df_add_errorTotalTime.iloc[:,
                                                   0]                /
df_add_errorTotalTime.iloc[:,

1]
    # 筛选易混淆词
    df_common_confused_words = df_add_errorTotalTime[
        df_add_errorTotalTime['error_rate'] > 0.2]
```

```
    df_common_confused_words

    # 查询易混淆词的等级
    import pandas as pd
    df  =  pd.read_excel('/Users/kangzhengwei/Desktop/HSK-2012.xlsx', header=None)

    # 获取易混淆词索引，即易混淆词
    a = df_common_confused_words.index
    # 这就对了
    df6 = pd.DataFrame()
    for i in a:
        # 暂时先不考虑不在HSK等级词汇表里的词汇了
        df7 = df[df.iloc[:, 0].str.contains(i) == True]
        df6 = pd.concat([df6, df7], axis=0)  # axis=0在y轴的方向上进行合并
    # 取消索引
    df6 = df6.reset_index()
    # 删除index列
    df6 = df6.drop('index', axis=1)
    df6

    # 把目标词的易混淆词写入到文件
    df6.to_csv('/Users/kangzhengwei/Desktop/新实验/目标词及其易混淆词/' + select_word + '.csv')
```

附录 H 词汇例句抽取

```
import re
import pandas as pd
from pandas import DataFrame
import jieba

f = open('/Users/kangzhengwei/Desktop/课文语料/所有的中级教材语料.txt', 'r')
element = f.read()



def seg_tail_split(str1, sep=r"。|？|！"):  # 分隔符可为多样的正则表达式
    # 保留分割符号，置于句尾，比如标点符号
    wlist = re.split(sep, str1)
    seg_word = re.findall(sep, str1)
    seg_word.extend(" ")  # 末尾插入一个空字符串，以保持长度和切割成分相同
    wlist = [x + y for x, y in zip(wlist, seg_word)]  # 顺序可根据需求调换
    return wlist



split_article_content = seg_tail_split(element)
print(split_article_content)

# # 部分四级词汇
# words_freq = pd.read_excel(
#                     '/Users/kangzhengwei/Desktop/新实验/last_grade4_word_freq.xlsx')

# 高频词
# search_word = words_freq.iloc[:, 1].values[:60]
```

```
    # 四级中频词、几个易混淆词、低频词
    search_word = ['看法', '深', '降低', '重视', '缺点', '气候','关键', '难受', '正式', '号码', '生意', '部分','想法', '大', '下降', '重要', '坏处', '天气','问题', '难过', '学期', '约会', '误会', '孙子','首都', '卫生间', '简直', '暖和', '景色', '堵车','语法', '迷路']

    print(search_word)

    data = pd.DataFrame(columns=['词汇', '句子'])

    # 下面这两个循环的顺序就对了
    for word in search_word:
        for i in split_article_content:
            # 判断词汇是不是在句子的词汇中，而不能只是是否出现在句子中。
            # 词频的统计是直接用结巴分词，而这个是先分割好句子，再看词汇有没有在句子中出现过。当然是不同的，因为有的句子中可能出现了两次这个词汇，那么词汇就显得多一些。这是非常合理的情况。
            if word in jieba.lcut(i):

                # 用 pandas 处理更方便一些
                #             print(word, i)
                new = pd.DataFrame({'词汇': word, '句子': i}, index=[0])
                #             print(new)
                data = data.append(new, ignore_index=True)
```

附录 I 句子分词，添加分词列、新分词列，筛选仅含有目标词汇及一级到三级词汇的句子

```
# 对句子进行分词并添加分词列
data['分词'] = data['句子'].apply(jieba.lcut)
data

# 把目标词汇添加到分词中，作为新分词列
def str_to_list(x):
    list1 = []
    list1.append(x) # 关键的步骤。其实就是列表中可以添加任何东西。这里是把字符串添加进了列表。
    return list1


data['新分词'] = data['词汇'].apply(str_to_list) + data['分词'] # 列表相加
data

# 筛选只包含目标词汇与新 HSK 一级到三级词汇的句子
# HSK1 级- HSK3 级的词汇

import pandas as pd

df = pd.read_excel('/Users/kangzhengwei/Desktop/HSK-2012.xls', header=None)
df1 = df[df.iloc[:, 0].str.contains('一级')]
df2 = df[df.iloc[:, 0].str.contains('二级')]
df3 = df[df.iloc[:, 0].str.contains('三级')]

low_grade_vocabulary = pd.concat([df1, df2, df3])
# low_grade_vocabulary
low_grade_vocabulary['等级'] = low_grade_vocabulary.iloc[:,
```

```
0].str[-3:-1]
    low_grade_vocabulary['词汇'] = low_grade_vocabulary.iloc[:,
0].str[:-4]
    len(low_grade_vocabulary)
    # low_grade_vocabulary

    low_grade_words = list(low_grade_vocabulary['词汇'].values)
    print(low_grade_words)
    HSK1_to_HSK3 = low_grade_words

    def right_sentence(x):

        punctuation = [',', '。', '?', '!', '”', '”', ':', '、',
'; ']
        for punc in punctuation:
            if punc in x:
                x.remove(punc)
        result = 1
        for i in x:
            if i in HSK1_to_HSK3 or i == data['新分词'][0]:
                j = 1
            else:
                j = 0
            result = result * j
        return result


    data["True_or_False"] = data['新分词'].apply(right_sentence)
    data[data["True_or_False"] == 1]
    # 这次竟然有两个这样的句子，但是太少了。因此这种方式还是不能奏效。
```

附录 J 对 HSK 常用词例句文件进行例句筛选实验

```
    # 抽取例句中的中文语料
    import re

    str1 = open('/Users/kangzhengwei/Desktop/新实验/HSK 常用词例句.txt', 'r').read()
    t = re.findall(

'[\u3002\uff1b\uff0c\uff1a\u201c\u201d\uff08\uff09\u3001\uff1f\u300a\u300b\u4e00-\u9fa5]',
        str1)
    content = ''.join(t)

    with open('/Users/kangzhengwei/Desktop/新实验/HSK 常用词例句_无空格.txt','w') as f:
        f.write(content)

    # 抽取部分词汇对应的中文句子
    import re
    import pandas
    from pandas import DataFrame
    import jieba

    f = open('/Users/kangzhengwei/Desktop/新实验/HSK 常用词例句_无空格.txt', 'r')
    element = f.read()



    def seg_tail_split(str1, sep=r"。|？|！"): # 分隔符可为多样的正则表达式
        # 保留分割符号，置于句尾，比如标点符号
        wlist = re.split(sep, str1)
        seg_word = re.findall(sep, str1)
```

```
        seg_word.extend(" ")  # 末尾插入一个空字符串，以保持长度和切割成分相同
        wlist = [x + y for x, y in zip(wlist, seg_word)]  # 顺序可根据需求调换
        return wlist


    split_article_content = seg_tail_split(element)
    print(split_article_content)

    # 部分四级词汇
    words_freq = pd.read_excel(
        '/Users/kangzhengwei/Desktop/新实验/last_grade4_word_freq.xlsx')

    search_word = words_freq.iloc[:, 1].values[:500]

    print(search_word)

    data = pd.DataFrame(columns=['词汇', '句子'])

    # 两个循环
    for word in search_word:
        for i in split_article_content:
            # 判断词汇是不是在句子的词汇中，而不能只是是否出现在句子中。
            # 词频的统计是直接用结巴分词，而这个是先分割好句子，再看词汇有没有在句子中出现过。
            if word in jieba.lcut(i):

                # 使用用pandas处理
                #             print(word, i)
                new = pd.DataFrame({'词汇': word, '句子': i}, index=[0])
```

```
                #               print(new)
                data = data.append(new, ignore_index=True)
    data

    # 筛选只包含目标词汇及新 HSK 一级到三级词汇的句子
    data['分词'] = data['句子'].apply(jieba.lcut)

    # 新分词

    def str_to_list(x):
        list1 = []
        list1.append(x) # 关键的步骤。其实就是列表中可以添加任何东西。这里是把字符串添加进了列表。
        return list1


    data['新分词'] = data['词汇'].apply(str_to_list) + data['分词'] # 列表相加
    data

    # HSK1、HSK2、HSK3 见附录 H
    HSK1_to_HSK3 = HSK1 + HSK2 +HSK3

    def right_sentence(x):

        punctuation = [',', '。', '?', '!', '”', '”', ':', '、', '; ']
        for punc in punctuation:
            if punc in x:
                x.remove(punc)
        result = 1
        for i in x:
            if i in HSK1_to_HSK3 or i == data['新分词'][0]:
                j = 1
```

```
        else:
            j = 0
        result = result * j
    return result


data["True_or_False"] = data['新分词'].apply(right_sentence)
data[data["True_or_False"]==1]
# 结果为空。
```

附录 K 添加句子中高等级词汇个数，标注出现的高等级词汇

```
# 前面需要保存为.pickle 格式，这样分词列与新分词列的每行内容才是列表，如果保存为.xls，每行内容则成为字符串了。
data = pd.read_pickle('/Users/kangzhengwei/Desktop/新实验/新分词.pickle')
data

# 四五六级词汇 选择

import pandas as pd

df = pd.read_excel('/Users/kangzhengwei/Desktop/HSK-2012.xls', header=None)
df4 = df[df.iloc[:, 0].str.contains('四级')]
df5 = df[df.iloc[:, 0].str.contains('五级')]
df6 = df[df.iloc[:, 0].str.contains('六级')]

high_grade_vocabulary = pd.concat([df4, df5, df6])
# high_grade_vocabulary
high_grade_vocabulary['等级'] = high_grade_vocabulary.iloc[:, 0].str[-3:-1]
high_grade_vocabulary['词汇'] = high_grade_vocabulary.iloc[:, 0].str[:-4]
len(high_grade_vocabulary)
# high_grade_vocabulary

high_grade_words = list(high_grade_vocabulary['词汇'].values)
print(high_grade_words)

# 统计每个句子的高等级词汇出现次数
def high_grade_words_count(x):
```

```
        result = 0
        for i in x:
            if i in high_grade_words:
                result += 1

        return result


    data["high_grade_words_count"] = data['分词
'].apply(high_grade_words_count)
    data

    # 统计每个句子中出现的高等级词汇
    def high_grade_words_appear(x):

        result = []
        for i in x:
            if i in high_grade_words:
                result.append(i)

        return result


    data["high_grade_words_appear"] = data['分词'].apply(
        high_grade_words_appear)    #  函数还是会作用于所选列的每一
行～～
    data
    # 保存为.pickle 文件
    data.to_pickle('/Users/kangzhengwei/Desktop/新实验/句子_加
high_grade_words_appear.pickle')
```

附录 L 获取词汇复杂度字典

```
    # 读取 BCC 语料库词频
    import pandas as pd

    word_freq = pd.read_csv('./global_wordfreq.txt', sep='\t',
header=None)
    word_freq

    from sklearn import preprocessing
    import numpy as np
    import math

    # 取对数后再取倒数
    # 利用 to_frame()将 Series 转换为 DataFrame

    x = word_freq.iloc[:, 1].apply(lambda x: math.log(x)).apply(
        lambda x: 1 / x).to_frame()
    # 将数据进行[0,1]规范化
    min_max_scaler = preprocessing.MinMaxScaler()
    minmax_x = min_max_scaler.fit_transform(x)
    print(minmax_x)

    # 将词汇难度添加到词频表
    word_complexity_with_log_reciprocal_MinMaxScale = pd.concat([
        word_freq, pd.DataFrame(minmax_x,
columns=['word_complexity_log_reciprocal_MinMaxScale'])],
axis=1,join='inner')

    word_complexity_with_log_reciprocal_MinMaxScale

    # 修改列名
    word_complexity_with_log_reciprocal_MinMaxScale.rename(columns={
        0:
```

```
        'words',
        1:
        'word_freq',
        'word_complexity_log_reciprocal_MinMaxScale':
        'word_complexity_log_reciprocal_MinMaxScale'
    }, inplace=True)
    word_complexity_with_log_reciprocal_MinMaxScale

    # 特征选择
    word_complexity_with_log_reciprocal_MinMaxScale =
word_complexity_with_log_reciprocal_MinMaxScale[
        ['words', 'word_complexity_log_reciprocal_MinMaxScale']]
    word_complexity_with_log_reciprocal_MinMaxScale

    # 写入 csv 文件
    word_complexity_with_log_reciprocal_MinMaxScale.to_csv(
        '/Users/kangzhengwei/Desktop/新实验/最终版
/word_complexity_with_log_reciprocal_MinMaxScale.csv',index=False
    )

    # 获取词汇复杂度字典
    import csv


    def csv2dict(in_file, key, value):
        new_dict = {}
        with open(in_file, 'r') as f:
            reader = csv.reader(f, delimiter=',')
            fieldnames = next(reader)
            reader = csv.DictReader(f, fieldnames=fieldnames,
delimiter=',')
            for row in reader:
                new_dict[row[key]] = row[value]
        return new_dict
```

```
    word_complexity_dict = csv2dict(
        '/Users/kangzhengwei/Desktop/新实验/最终版
/word_complexity_with_log_reciprocal_MinMaxScale.csv',
        'words', 'word_complexity_log_reciprocal_MinMaxScale')
    word_complexity_dict
```

附录 M 句子复杂度计算

```
    data = pd.read_pickle('/Users/kangzhengwei/Desktop/新实验/最终版/句子_加high_grade_words_appear.pickle')
    data

    data = data.iloc[:,1:]
    data

    def calculate_sentence_complexity(x):
        # 去除列表中的中文标点符号
        punctuation = [',', '。', '?', '!', '”', '”', ':', '、', ';']
        for punc in punctuation:
            if punc in x:
                x.remove(punc)

        # 计算句子复杂度
        result = 0
        for i in x:

            # 对应词汇复杂度的字典
            j = float(word_complexity_dict.get(i, 0))
            #         print(type(j))

            result = result + j  # 认识到了这儿有bug。这次修改过来了。
        return result


    data['sentence_complexity'] = data['分词'].apply(calculate_sentence_complexity)
    data
```

```
    # 对句子复杂度进行标准化

    from sklearn import preprocessing
    import numpy as np

    x = data['sentence_complexity'].to_frame()
    # 将数据进行[0,1]规范化
    min_max_scaler = preprocessing.MinMaxScaler()
    minmax_x = min_max_scaler.fit_transform(x)
    print(minmax_x)

    # 将规范化后的句子复杂度添加到 data 上

    data_with_sentence_complexity_scale = pd.concat([
        data,
        pd.DataFrame(minmax_x,
columns=['sentence_complexity_MinMaxScale'])
    ],
                                                    axis=1,
                                                    join='inner')
    data_with_sentence_complexity_scale

    # 按 词汇 进行分组，并按照句子复杂度（经过标准化）与高等级词汇的
数量进行升序排列

    data_with_sentence_complexity_scale =
data_with_sentence_complexity_scale.groupby(
        '词汇').apply(lambda x: x.sort_values(
            by=['sentence_complexity_MinMaxScale',
'high_grade_words_count'],
            ascending=[True, True]))
    data_with_sentence_complexity_scale

    # 写入文件
```

```
    data_with_sentence_complexity_scale.to_pickle(
        '/Users/kangzhengwei/Desktop/新实验/最终版
/data_with_sentence_complexity_scale.pickle'
    )

    # 写入文件
    data_with_sentence_complexity_scale.to_excel(
        '/Users/kangzhengwei/Desktop/新实验/最终版
/data_with_sentence_complexity_scale.xlsx'
    )
```

附录 N 词汇搭配提取

```
# 读取 excel 文件
select_word = '通过'
import pandas as pd

data = pd.read_excel('/Users/kangzhengwei/Desktop/新实验/搭配信息获取/' + select_word + '.xlsx')
data

# 获取第二列
df1 = data.iloc[:, 1]
df1

# 使用正则表达式提取出搭配（去除目标词和空格）
# 去除目标词
df1.str.extract(r'([^(' + select_word + ')].*[^(' + select_word + ')])', expand=False)

# 提取词汇搭配
df2 = df1.str.extract(r'([^(' + select_word + ')].*[^(' + select_word + ')])', expand=False).str.extract(r'([\u4e00-\u9fa5]+)') # 正好可以把‘开放 的’中的‘ 的’去掉。
df2

# 识别词汇搭配及其词汇等级
df3 = pd.read_excel('/Users/kangzhengwei/Desktop/HSK-2012.xlsx', header=None)
df3

import numpy as np


def word_and_level(x):
```

```
        for i in df3.iloc[:, 0].values:
            if x is not np.nan:  # 这里还必须用 is not，不能用!=
                if x in i: # 比如’服务‘ in ’服务员‘,’服务员‘也可以提取出来。
                    return i
            else:
                return None


    df4 = df2.iloc[:, 0].apply(word_and_level)
    df4

    # 把等级列添加到 data 中
    data['word_grade'] = df4
    data

    # word_grade 列过滤空值
    data2 = data[data['word_grade'].isnull() == False]
    data2

    # 筛选新 HSK 一级到三级词汇，按照互信息降序排列
    data3  =  data2[data2['word_grade'].str.contains('三级|二级|一级')].sort_values(
        by='Mutual Information', ascending=False)
    data3

    # 写入到文件
    data3.to_csv('/Users/kangzhengwei/Desktop/新实验/搭配信息_添加词汇等级/'+select_word+'.csv')
```

附录 0 字词发音文件爬取

```
# 百度翻译音频爬取
import requests

url = "https://fanyi.baidu.com/gettts"

with open(
        '/Users/kangzhengwei/Desktop/ 百 度 翻 译 音 频 文 件 /words_collocates_sentences_2.txt',
        'r') as f:

    my_list = f.readlines()

    print(my_list)

    for text in my_list:

        text = text.strip()

        print('你刚才输入的是：{}'.format(text))

        querystring = {"lan": "zh", "spd": "5", "source": "web"}
        querystring["text"] = text
        payload = ""
        headers = {
            'User-Agent':
            'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.84 Safari/537.36'
        }

        response = requests.request("GET",
                                    url,
```

```
                                        data=payload,
                                        headers=headers,
                                        params=querystring)
            r = response.content
            # print(response.text)
            fo = open('/Users/kangzhengwei/Desktop/百度翻译音频文件/{}.mp3'.format(text),
                      'wb')  # 注意要用'wb',b 表示二进制，不要用'w'
            fo.write(r)  # r.content -> requests 中的二进制响应内容：以字节的方式访问请求响应体，对于非文本请求
            fo.close()
            print('写入成功 ')
```

附录 P 将 mp3 格式音频转换为 wav 格式

```
import os
from os import path
from pydub import AudioSegment
for info in os.listdir('/Users/kangzhengwei/Desktop/百度翻译音频文件'):
    domain = os.path.abspath('/Users/kangzhengwei/Desktop/百度翻译音频文件')
    complete_path = os.path.join(domain, info)  #将路径与文件名结合起来就是每个文件的完整路径
    if complete_path[-4:] == '.mp3':
        print(complete_path)

        try:

            sound = AudioSegment.from_mp3(complete_path)
            sound.export(complete_path[:-4] + '.wav', format='wav')
        except:
            print('a little mistake')
```