最近因为学校项目的原因，开始从爬虫接触到自然语言处理，这方面市面上的教程还是比较少的，很多时候网上也找不到代码或者教程，所以我打算写一篇笔记来供大家或者以后复习使用。

环境问题

- Java环境是需要的
- python环境
- 各种库

由于环境问题不是重要的问题，重要的还是一些代码的解读和使用(对我这种新手来说是的，毕竟先学会使用才有兴趣了解底层嘛)，我们直接进入代码解读，不过事先提醒一下，环境问题还是比较复杂的。

Hanlp代码解读

Hanlp分词

- 我们需要运用jpype库，这个库我们输入如下语句就可以实现了

```
pip install jpype1
```

我们需要用到Java的代码，所以我们需要在Pyhton中启动Java的虚拟机，启动代码如下

```
startJVM(getDefaultJVMPath(),
         "-
Djava.class.path=C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-
1.7.7-release\\hanlp-1.7.7-release\\hanlp-
1.7.7.jar;C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-1.7.7-
release\\hanlp-1.7.7-release",
        "-Xms1g",
        "-Xmx1g")  # 启动JVM，Linux需替换分号;为冒号:

# Djava.class.path是你下载hanlp中java包的路径+文件名
# 后面还有跟上这个路径
# 这一步是用来启动JAVA的虚拟机
```

这些步骤做完之后我们就可以在Pyhton中调用Java的代码了

- 先导入Java的类，导入方法如下

```
对象 = JClass('Java类名')
```

我们下面来细将一下各个方法的实现代码

Hanlp分词

```
HanLP = JClass('com.hankcs.hanlp.HanLP')
# 中文分词
print(HanLP.segment('你好，欢迎在Python中调用HanLP的API'))
print("-" * 70)

# ouput
==============================HanLP分词==============================
[你好/vl，，/w，欢迎/v，在/p，Python/nx，中/f，调用/v，HanLP/nx，的/ude1，API/nx]
----------------------------------------------------------------------
```

我们可以很清楚的看见，我们将词语分出来，然后会有/来分割他们，右面呢就是词性，在本文的最后会附上Hanlp的词性表，在以后会有很大的用处。

标准分词

```
StandardTokenizer = JClass('com.hankcs.hanlp.tokenizer.StandardTokenizer')
print("=" * 30 + "标准分词" + "=" * 30)
print(StandardTokenizer.segment('你好，欢迎在Python中调用HanLP的API'))
print("-" * 70)

# ouput
==============================标准分词==============================
[你好/vl，，/w，欢迎/v，在/p，Python/nx，中/f，调用/v，HanLP/nx，的/ude1，API/nx]
----------------------------------------------------------------------
```

NLP分词

- NLP分词NLPTokenizer会执行全部命名实体识别和词性标注

```
print("=" * 30 + "NLP分词" + "=" * 30)
NLPTokenizer = JClass('com.hankcs.hanlp.tokenizer.NLPTokenizer')
print(NLPTokenizer.segment('中国科学院计算技术研究所的宗成庆教授正在教授自然语言处理课程'))
print("-" * 70)

# output
==============================NLP分词==============================
[中国科学院计算技术研究所/nt, 的/u, 宗/q, 成庆/vn, 教授/n, 正在/d, 教授/n, 自然语言处理/nz, 课程/n]
----------------------------------------------------------------------
```

索引分词

- 索引分词会将分出来的词的位置分出来

```
print("=" * 30 + "索引分词" + "=" * 30)
IndexTokenizer = JClass('com.hankcs.hanlp.tokenizer.IndexTokenizer')
termList = IndexTokenizer.segment("主副食品");
for term in termList:
    print(str(term) + " [" + str(term.offset) + ":" + str(term.offset + len(term.word)) + "]")
print("-" * 70)

# output
==============================索引分词==============================
主副食品/n [0:4]
主副食/j [0:3]
副食品/n [1:4]
副食/n [1:3]
食品/n [2:4]
----------------------------------------------------------------------
```

极速词典分词

- 特点 快！！！

```
print("=" * 30 + " 极速词典分词" + "=" * 30)
SpeedTokenizer = JClass('com.hankcs.hanlp.tokenizer.SpeedTokenizer')
print(NLPTokenizer.segment('江西鄱阳湖干枯，中国最大淡水湖变成大草原'))
print("-" * 70)

# output
============================== 极速词典分词==============================
[江西鄱阳湖干枯/nt, ，/w, 中国/n, 最大/a, 淡水湖/n, 变成/v, 大/a, 草原/n]
----------------------------------------------------------------------
```

自定义分词

```
print("=" * 30 + " 自定义分词" + "=" * 30)
CustomDictionary = JClass('com.hankcs.hanlp.dictionary.CustomDictionary')
CustomDictionary.add('攻城狮')
CustomDictionary.add('单身狗')
HanLP = JClass('com.hankcs.hanlp.HanLP')
print(HanLP.segment('攻城狮逆袭单身狗，迎娶白富美，走上人生巅峰'))
print("-" * 70)

# output
============================== 自定义分词==============================
[攻城狮/nz，逆袭/nz，单身狗/nz，，/w，迎娶/v，白富美/nr，，/w，走上/v，人生/n，巅峰/n]
----------------------------------------------------------------------
```

命名实体识别与词性标注

```
print("=" * 20 + "命名实体识别与词性标注" + "=" * 30)
NLPTokenizer = JClass('com.hankcs.hanlp.tokenizer.NLPTokenizer')
print(NLPTokenizer.segment('中国科学院计算技术研究所的宗成庆教授正在教授自然语言处理课程'))
print("-" * 70)

# output
====================命名实体识别与词性标注==============================
[中国科学院计算技术研究所/nt，的/u，宗/q，成庆/vn，教授/n，正在/d，教授/n，自然语言处理/nz，课程/n]
----------------------------------------------------------------------
```

关键词提取

```
HanLP = JClass('com.hankcs.hanlp.HanLP')
document = "水利部水资源司司长陈明忠9月29日在国务院新闻办举行的新闻发布会上透露，" \
    "根据刚刚完成了水资源管理制度的考核，有部分省接近了红线的指标，" \
           "有部分省超过红线的指标。对一些超过红线的地方，陈明忠表示，对一些取用水项目进行区域的限批，" \
           "严格地进行水资源论证和取水许可的批准。"
print("=" * 30 + "关键词提取" + "=" * 30)
print(HanLP.extractKeyword(document, 8))
print("-" * 70)

print("=" * 30 + "自动摘要" + "=" * 30)
print(HanLP.extractSummary(document, 3))
print("-" * 70)
```

```
text = r"算法工程师\n 算法（Algorithm）是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一个算法的优劣可以用空间复杂度与时间复杂度来衡量。算法工程师就是利用算法处理事物的人。\n \n 1职位简介\n 算法工程师是一个非常高端的职位；\n 专业要求：计算机、电子、通信、数学等相关专业；\n 学历要求：本科及其以上的学历，大多数是硕士学历及其以上；\n 语言要求：英语要求是熟练，基本上能阅读国外专业书刊；\n 必须掌握计算机相关知识，熟练使用仿真工具MATLAB等，必须会一门编程语言。\n\n2研究方向\n 视频算法工程师、图像处理算法工程师、音频算法工程师 通信基带算法工程师\n \n 3目前国内外状况\n 目前国内从事算法研究的工程师不少，但是高级算法工程师却很少，是一个非常紧缺的专业工程师。算法工程师根据研究领域来分主要有音频/视频算法处理、图像技术方面的二维信息算法处理和通信物理层、雷达信号处理、生物医学信号处理等领域的一维信息算法处理。\n 在计算机音视频和图形图像技术等二维信息算法处理方面目前比较先进的视频处理算法：机器视觉成为此类算法研究的核心；另外还有2D转3D算法(2D-to-3D conversion)，去隔行算法(de-interlacing)，运动估计运动补偿算法(Motion estimation/Motion Compensation)，去噪算法(Noise Reduction)，缩放算法(scaling)，锐化处理算法(Sharpness)，超分辨率算法(Super Resolution),手势识别(gesture recognition),人脸识别(face recognition)。\n 在通信物理层等一维信息领域目前常用的算法：无线领域的RRM、RTT，传送领域的调制解调、信道均衡、信号检测、网络优化、信号分解等。\n 另外数据挖掘、互联网搜索算法也成为当今的热门方向。\n"
print("=" * 30 + "短语提取" + "=" * 30)

print(HanLP.extractPhrase(text, 10))
print("-" * 70)

# output
==============================关键词提取==============================
[水资源，陈明忠，进行，红线，部分，项目，用水，国务院新闻办]
----------------------------------------------------------------------

==============================自动摘要==============================
[水利部水资源司司长陈明忠9月29日在国务院新闻办举行的新闻发布会上透露，严格地进行水资源论证和取水许可的批准，有部分省超过红线的指标]
----------------------------------------------------------------------

==============================短语提取==============================
[信息算法处理，一维信息，通信物理层，互联网搜索算法，从事算法研究，信号处理生物医学，信号检测网络优化，先进视频处理，利用算法处理，图像处理算法工程师音频算法工程师]
----------------------------------------------------------------------
```


准确分词（自定义字典）

在我们进行分词的时候，我们可能会遇见我们一些词并不想被分开，我们该如何解决这个问题呢?

我们的文本是一段有关于医疗方面的文本

```
恶性肿瘤的分期越高，患者预后越差。通过对肿瘤不同恶性程度的划分，TNM分期在预测预后方面不断完善。
但是有国外研究发现，部分结肠癌的预后并不能完全按照一般的分期阶梯进行预测。
TNM分期不太能明确地区分||期和III期结肠癌患者的预后ii期，特别是在接受辅助化疗的患者中，他们的5年总生存期在50.1%-9.8%。
此外已知影响结肠癌生存的患者或疾病特征，包括年龄、性别、原发疾病位置、肿瘤分级、阳性淋巴结数目（LNs）、接受检查的
LNs数目、淋巴管和周围神经浸润、肠道梗阻或穿孔、以及辅助治疗（氟尿嘧啶单药或联合奥沙利铂），并未直接纳入TNM分期系统。
在多变量模型中，分子标记的微卫星不稳定性（MSI）和BRAF或KRAS基因突变联合详细的临床病理学诠释可以多大程度改善预后评估
```

目前尚不清楚。近期，发表在Annals of oncology杂志上的一项回顾性研究，在TNM分期系统基础上联合汇集的标志物对II期和III期
结肠癌总生存期进行预测。
研究人员将缺失随机数据插补后，利用3期辅助化疗试验（n=3016）–N0147（NCT00079274）和PETACC3（NCT00026273）–产生的患者亚组
聚集构建了一个5年总生存期多变量Cox模型，随后在剩余的临床试验样本（n=1499）中进行内部验证，并且还在不同人群队列中外部分析，
包括接受化疗（n=949）或者未接受化疗（n=1080）的结肠癌患者，以及没有治疗注释患者。
研究分析发现：
在根据临床试验队列和观察性研究做出的多变量模型中，TMN分期，MSI和BRAFV600E基因突变状态仍然是独立预后因素。
单纯TNM模型的一致性指数（Concordance-indices）为0.61-0.68，而增加分子标记物、临床病理特征和所有协变量后的
一致性指数分别增加至0.63-0.71、0.65-0.73和0.66-0.74。
在有完整注释的验证队列中，单独TNM的综合时间依赖AUC值为0.64，纳入临床病理特征联合或不联合分子标记物的AUC增加为0.67。
在接受辅助化疗的患者队列中，通过TNM、临床病理特征和分子标记物的方差（R2）相对比例平均值分别为65%、25%和10%。
因此，将MSI、BRAFV600E和KRAS基因突变状态纳入TNM分期系统的总生存期模型可以提高精确预测II期和III期结肠癌患者的能力，
而且包括临床病理特征的多变量模型中会增加预测准确性，特别是需要接受化疗的患者。

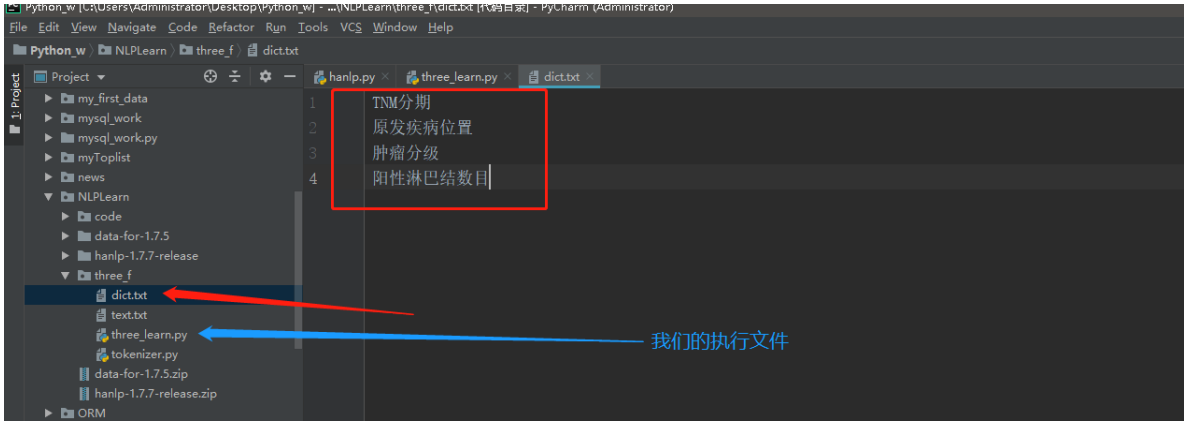
jieba分词

我们来看看jieba分词怎么操作

```
import jieba

with open("text.txt", encoding="utf8") as fp:
    lines = fp.readlines()

for line in lines:
    words = jieba.cut(line)
    res = " ".join(list(words))
    print(res)

# output (一部分)
恶性肿瘤 的 分期 越高 ， 患者 预后 越差 。 通过 对 肿瘤 不同 恶性 程度 的 划分 ， TNM 分期 在 预测 预后 方面 不断完善 。

但是 有 国外 研究 发现 ， 部分 结肠癌 的 预后 并 不能 完全 按照 一般 的 分期 阶梯 进行 预测 。

TNM 分期 不太能 明确 地区 分 | | 期 和 III 期 结肠癌 患者 的 预后 ii 期 ， 特别 是 在 接受 辅助 化疗 的 患者 中 ， 他们 的 5 年 总 生存期 在 50.1% - 9.8% 。

此外 已知 影响 结肠癌 生存 的 患者 或 疾病 特征 ， 包括 年龄 、 性别 、 原发 疾病 位置 、 肿瘤 分级 、 阳性 淋巴结 数目 （ LNs ） 、 接受 检查 的
```

我们很明显的看见了一些词语是被分开了的，但是呢假如我们需要原发 疾病 位置，肿瘤分级，阳性淋巴结数目，TNM 分期，这些词语不分开，我们应该怎么做呢?

我们在同目录下创建一个dict.txt 然后在里面写上你不想被分开的词语，我们来通过图片来看看



然后在程序中怎么加载呢?

- 第一种方式，使用jieba.load_userdict()

```
import jieba

jieba.load_userdict("dict.txt")

with open("text.txt", encoding="utf8") as fp:
    lines = fp.readlines()

for line in lines:
    words = jieba.cut(line)
    res = " ".join(list(words))
    print(res)

# output (一部分)
恶性肿瘤 的 分期 越高 ， 患者 预后 越差 。 通过 对 肿瘤 不同 恶性 程度 的 划分 ， TNM分期 在 预测 预后 方面 不断完善 。

但是 有 国外 研究 发现 ， 部分 结肠癌 的 预后 并 不能 完全 按照 一般 的 分期 阶梯 进行 预测 。

TNM分期 不太能 明确 地区 分 | | 期 和 III 期 结肠癌 患者 的 预后 ii 期 ， 特别 是 在 接受 辅助 化疗 的 患者 中 ， 他们 的 5 年 总 生存期 在 50.1% - 9.8% 。

此外 已知 影响 结肠癌 生存 的 患者 或 疾病 特征 ， 包括 年龄 、 性别 、 原发疾病位置 、 肿瘤分级 、 阳性淋巴结数目 （ LNs ） 、 接受 检查 的
```

hanlp分词添加字典

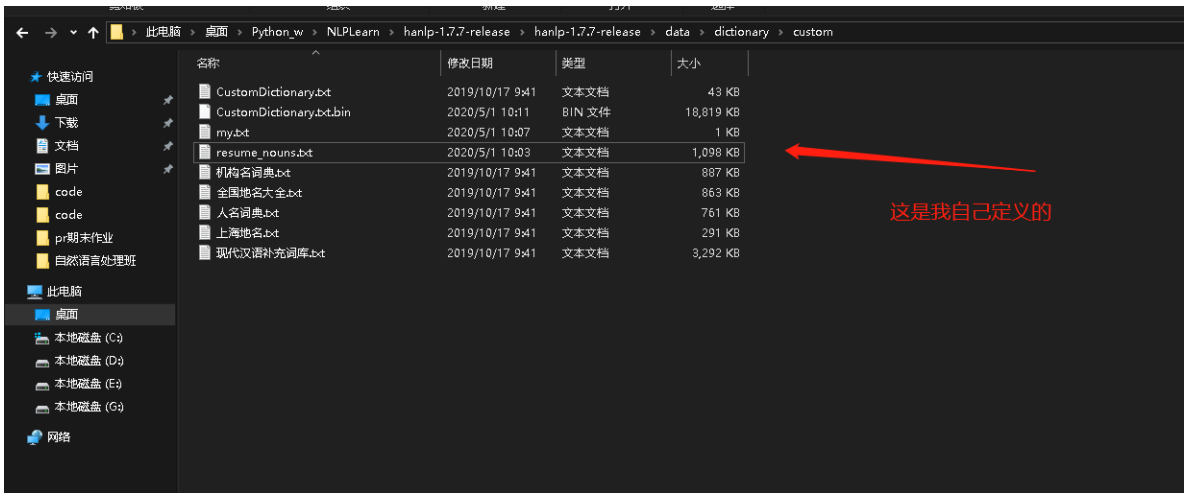
首先我们按照NLP分词的方式来分一下词，看看跟jieba有什么区别

```
from jpype import *

startJVM(getDefaultJVMPath(),
         "-
Djava.class.path=C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-
1.7.7-release\\hanlp-1.7.7-release\\hanlp-
1.7.7.jar;C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-1.7.7-
release\\hanlp-1.7.7-release",
        "-Xms1g",
        "-Xmx1g")
```

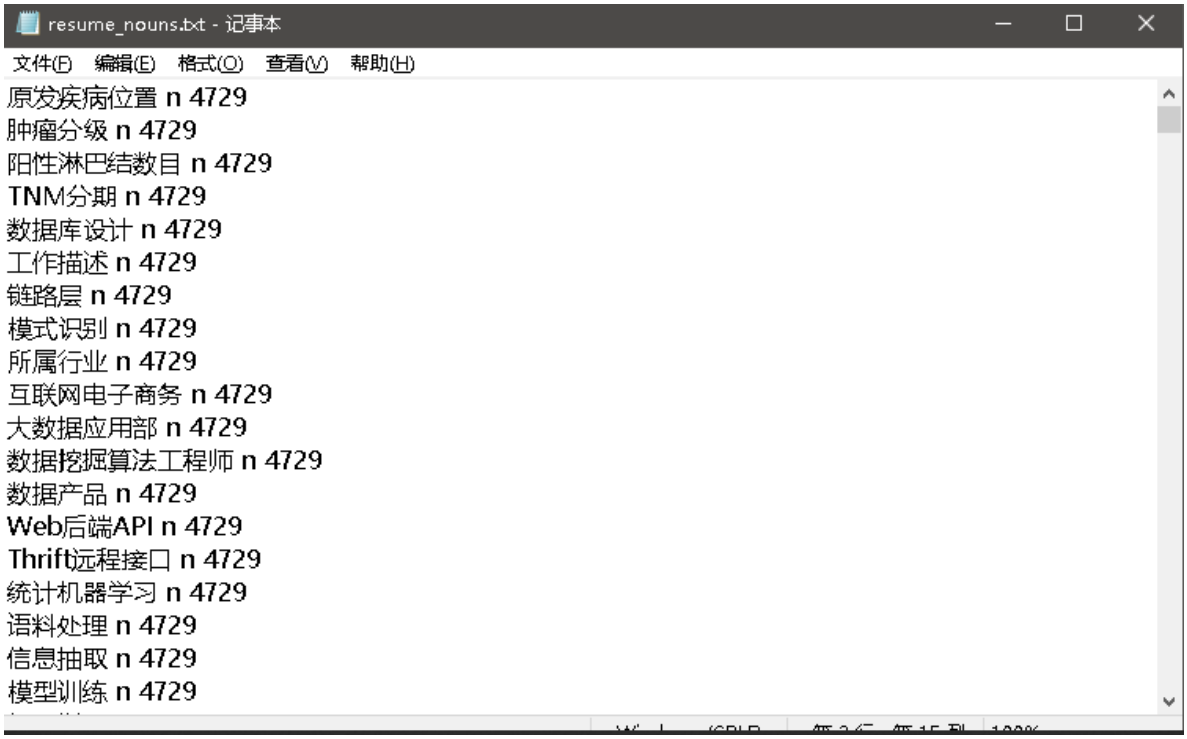
```
hanlp_tool = JClass('com.hankcs.hanlp.tokenizer.StandardTokenizer')

with open("text.txt", encoding="utf8") as fp:
    lines = fp.readlines()

for line in lines:
    if len(line) > 0:
        res = " ".join([word_pos_item.toString().split('/')[0] for word_pos_item
in hanlp_tool.segment(line)])
        print(res)

shutdownJVM()

# output
恶性肿瘤 的 分期 越高 ， 患者 预后 越差 。 通过 对 肿瘤 不同 恶性 程度 的 划分 ， TNM 分期
在 预测 预后 方面 不断完善 。

但是 有 国外 研究 发现 ， 部分 结肠癌 的 预后 并 不能 完全 按照 一般 的 分期 阶梯 进行 预测
。

TNM 分期 不太能 明确 地 区分 || 期 和 III 期 结肠癌 患者 的 预后 ii 期 ， 特别是在 接受
辅助 化疗 的 患者 中 ， 他们 的 5 年 总 生存期 在 50.1 %- 9.8 % 。

此外 已知 影响 结肠癌 生存 的 患者 或 疾病 特征 ， 包括 年龄 、 性别 、 原发 疾病 位置 、
肿瘤 分级 、 阳性 淋巴结 数目 （ LNs ） 、 接受 检查 的
```

看起来分出的也和jieba差不多，那我们看看他怎么用自己自用的字典呢?

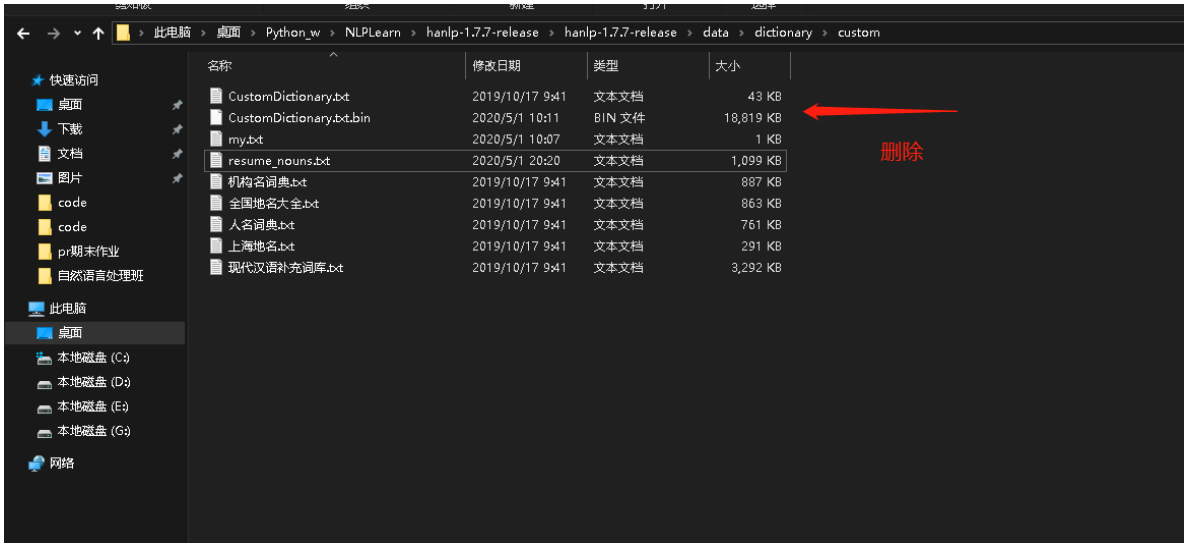
我们在Hanlp的data文件夹下有她分词的数据集，我们可以加入在里面，具体是这样的



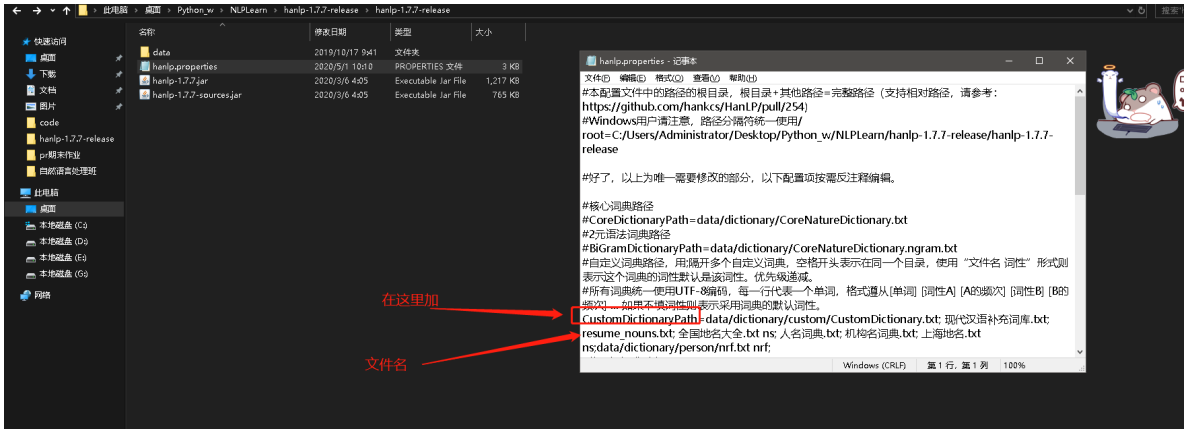
我们看看里面的内容是怎么样的



ok 我们保存之后也把这个文件删除，如下图



之后再下面这个文件中加入这个自定义字典的文件名称



ok 那我们重新运行一下，代码是不用改的，一下只有一部分运行结果

```
恶性肿瘤 的 分期 越高 ， 患者 预后 越差 。 通过 对 肿瘤 不同 恶性 程度 的 划分 ， TNM分期 在 预测 预后 方面 不断完善 。

但是 有 国外 研究 发现 ， 部分 结肠癌 的 预后 并 不能 完全 按照 一般 的 分期 阶梯 进行 预测 。

TNM分期 不太能 明确 地 区分 || 期 和 III 期 结肠癌 患者 的 预后 ii 期 ， 特别是在 接受 辅助 化疗 的 患者 中 ， 他们 的 5 年 总 生存期 在 50.1 %- 9.8 % 。

此外 已知 影响 结肠癌 生存 的 患者 或 疾病 特征 ， 包括 年龄 、 性别 、 原发疾病位置 、 肿瘤分级 、 阳性淋巴结数目 （ LNs ） 、 接受 检查 的
```

ok 很明显能够看出来是成功了的。

以上就是自定义字典的过程了

词性标注代码实现及信息提取

文本：

```
自我的评价

本人诚实正直，对工作认真负责，吃苦耐劳，善于创新，敢于迎接挑战及承担责任，富有工作热情，乐业敬业，善于与人沟通。营造和谐的工作氛围，注重人性化管理，能带动下属充分发挥团队合作精神，为公司创造效益！

求职意向
到岗时间：一个月之内
工作性质：全职
希望行业：通信/电信运营
目标地点：广州
期望月薪：面议/月
目标职能：数据分析专员
工作经验
2014/11 – 2015/9：XX有限公司[10个月]
所属行业：通信/电信运营
数据部　　数据分析专员
1.　数据库日常简单维护，熟悉SQL查询语句。
2.　数据分析，协助客户定位网络疑问问题。
3.　投诉建模，通过匹配大量的投诉用户及其上网行为，分析其可能投诉的原因并进行建模。
2013/5 – 2014/10：XX有限公司[1年5个月]
所属行业：通信/电信运营
数据部　　数据分析专员
1.　日常办公用品采购，基站租赁合同处理及工程物资采购。
2.　ERP项目支出入账及物资装配，投诉工单处理，通信基站故障处理。
3.　按排会议室，会议记要记录及整理，公文编辑分发。
```

一下代码我写成了两个文件，第二个文件是存放着调用java代码的函数，我们来看看我是怎么实现的，文件名为：tokenizer.py

```
from jpype import *

startJVM(getDefaultJVMPath(),
```

```
        "-
Djava.class.path=C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-
1.7.7-release\\hanlp-1.7.7-release\\hanlp-
1.7.7.jar;C:\\Users\\Administrator\\Desktop\\Python_w\\NLPLearn\\hanlp-1.7.7-
release\\hanlp-1.7.7-release",
        "-Xms1g",
        "-Xmx1g")  # JPype 提供的 startJVM() 函数的作用是启动 JAVA 虚拟机，所以在后续
的任何 JAVA 代码被调用前，必须先调用此方法启动 JAVA 虚拟机。

Tokenizer = JClass('com.hankcs.hanlp.tokenizer.StandardTokenizer')


# """以下我们定义一些我们想要的词性和不想要的词性，这个操作会有助于我们的分析"""
# """词性表在最后面"""

keep_pos =
"q,qg,qt,qv,s,t,tg,g,gb,gbc,gc,gg,gm,gp,m,mg,Mg,mq,n,an,vn,ude1,nr,ns,nt,nz,nb,nb
a,nbc,nbp,nf,ng,nh,nhd,o,nz,nx,ntu,nts,nto,nth,ntch,ntcf,ntcb,ntc,nt,nsf,ns,nrj,n
rf,nr2,nr1,nr,nnt,nnd,nn,nmc,nm,nl,nit,nis,nic,ni,nhm,nhd"
keep_pos_nouns = set(keep_pos.split(","))

keep_pos_v = "v,vd,vg,vf,vl,vshi,vyou,vx,vi"
keep_pos_v = set(keep_pos_v.split(","))
keep_pos_p = set(['p', 'pbei', 'pba'])

# 这些是你不要的词性，我们都写在这里，当程序检测出词性是输入您这里写的您不想要的词性，那么程序
是自动过滤他
drop_pos_set = set(['xu', 'xx', 'y', 'yg', 'wh', 'wky', 'wkz', 'wp', 'pp', 'ws',
'wyy', 'wyz', 'wb', 'u', 'ud', 'ude1', 'ude2', 'ude3','udeng', 'udh'])

def to_string(sentence, return_generator=False):
    # 这个函数是看看调用的人是否需要将词性保留在里面，如果需要我们则使用else，反之是if
    if return_generator:
        return " ".join([word_pos_item.toString().split('/')[0] for word_pos_item
in Tokenizer.segment(sentence)])
    else:
        return [(word_pos_item.toString().split('/')[0],
word_pos_item.toString().split('/')[1]) for word_pos_item in
Tokenizer.segment(sentence)]


def seg_sentences(sentence, with_filter=True, return_generator=False):
    # 我们这个函数是判断哪些是你不要的词性，将您想要的筛选出来，不想要的就去掉，不要了
    segs = to_string(sentence, return_generator=return_generator)
    if with_filter:
        g = [word_pos_pair[0] for word_pos_pair in segs if
             len(word_pos_pair) == 2 and word_pos_pair[0] != ' ' and
word_pos_pair[1] not in drop_pos_set]
    else:
        g = [word_pos_pair[0] for word_pos_pair in segs if len(word_pos_pair) ==
2 and word_pos_pair[0] != ' ']
    return iter(g) if return_generator else g
```

ok 那我们来看看我们的主程序是怎么写的吧。

```
from tokenizer import seg_sentences,to_string
```

```
with open("info.txt",encoding="utf8") as f:
    res = f.readlines()

for line in res:
    line = line.strip()  # 将换行符一些空格删除
    if len(line) > 0:
        strings = " ".join(seg_sentences(line))
        print("原本的为------------>",line)
        print("筛选过的分词为------>",strings)

# output(只有一部分)
原本的为------------> 自我的评价
筛选过的分词为------> 自我 评价
原本的为------------> 本人诚实正直，对工作认真负责，吃苦耐劳，善于创新，敢于迎接挑战及承担责任，富有工作热情，乐业敬业，善于与人沟通。营造和谐的工作氛围，注重人性化管理，能带动下属充分发挥团队合作精神，为公司创造效益！
筛选过的分词为------> 本人 诚实 正直 对 工作 认真 负责 吃苦耐劳 善于 创新 敢于 迎接挑战 及 承担责任 富有 工作 热情 乐业 敬业 善于 与 人 沟通 营造 和谐 工作 氛围 注重 人性化 管理 能 带动 下属 充分发挥 团队合作 精神 为 公司 创造 效益
原本的为------------> 求职意向
筛选过的分词为------> 求职意向
原本的为------------> 到岗时间：一个月之内
筛选过的分词为------> 到 岗 时间 一个 月 之内
原本的为------------> 工作性质：全职
筛选过的分词为------> 工作 性质 全职
原本的为------------> 希望行业：通信/电信运营
筛选过的分词为------> 希望 行业 通信 电信 运营
```

可以很明显的看见我们一些没有用的词语已经被我们筛选掉了，这样对我们的分析还是很有效果的

TextRank

这个算法和PageRank的算法比较相似，大家可以去了解一下他的算法是什么做成的，这里大概的说一下，不是很详细，就是大概理解一下，虽然别人是帮我们写好并封装起来了，但是还是要了解怎么实现的，具体的实现还是去官方看会更好一点

谈起自动摘要算法，常见的并且最易实现的当属TF-IDF，但是感觉TF-IDF效果一般，不如TextRank好。

TextRank是在Google的PageRank算法启发下，针对文本里的句子设计的权重算法，目标是自动摘要。它利用投票的原理，让每一个单词给它的邻居（术语称窗口）投赞成票，票的权重取决于自己的票数。这是一个“先有鸡还是先有蛋”的悖论，PageRank采用矩阵迭代收敛的方式解决了这个悖论。TextRank也不例外：

PageRank的计算公式：

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

正规的TextRank公式

正规的TextRank公式在PageRank的公式的基础上，引入了边的权值的概念，代表两个句子的相似度。

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

但是很明显我只想计算关键字，如果把一个单词视为一个句子的话，那么所有句子（单词）构成的边的权重都是0（没有交集，没有相似性），所以分子分母的权值w约掉了，算法退化为PageRank。所以说，这里称关键字提取算法为PageRank也不为过。

用途：

1.提取出文本中比较重要的关键词

2.选出一段话中，哪个关键词出现的次数多的问题

我们来看看jieba的实现

```
from jieba import analyse
textrank = analyse.textrank
text = "文/观察者网 徐乾昂】新冠肺炎疫情防控是对各国执政能力的一场测验。美国政府整体反应滞后，准备不足，导致疫情大爆发。截止目前美国新冠肺炎确诊病例累计已达1.3万多例，半个月内从“破百”升至“破万”。而美国政府拒不承认自身抗疫工作的不足，试图把“锅”甩给中国。3月19日的白宫记者会上，特朗普再次故意将“新冠病毒”妄称为“中国病毒”。而现场有《华盛顿邮报》的摄影记者发现，特朗普的演讲稿上明明标注的是“新冠”病毒，但却被划掉，写上了“中国”病毒。"
keyws = textrank(text)

print(keyws)

# output
['新冠', '肺炎', '病毒', '工作', '疫情', '准备', '美国', '中国', '测验', '发现', '整体', '反应', '截止', '确诊', '病例', '称为', '演讲稿', '防控', '爆发', '执政']
```

ok，此文本最重要的一些词语都被我们找出来，那我们能不能控制一些参数呢？当然可以，我们去看看。

```
    def textrank(self, sentence, topK=20, withWeight=False, allowPOS=('ns', 'n',
'vn', 'v'), withFlag=False):
        """
        Extract keywords from sentence using TextRank algorithm.
        Parameter:
            - topK: return how many top keywords. `None` for all possible words.
            - withWeight: if True, return a list of (word, weight);
                          if False, return a list of words.
            - allowPOS: the allowed POS list eg. ['ns', 'n', 'vn', 'v'].
                        if the POS of w is not in this list, it will be
filtered.
            - withFlag: if True, return a list of pair(word, weight) like
posseg.cut
                        if False, return a list of words
        """
       *****
```

这个jieba中textrank函数，我们看看他的参数topK就是选出多少个词，withWeigh是否需要显示权重，allowPOS选择的词性，withFlag=False是否显示出词性

ok，我们现在来调整一下

```
from jieba import analyse
textrank = analyse.textrank
text = "文/观察者网 徐乾昂】新冠肺炎疫情防控是对各国执政能力的一场测验。美国政府整体反应滞后，准备不足，导致疫情大爆发。截止目前美国新冠肺炎确诊病例累计已达1.3万多例，半个月内从“破百”升至“破万”。而美国政府拒不承认自身抗疫工作的不足，试图把“锅”甩给中国。3月19日的白宫记者会上，特朗普再次故意将“新冠病毒”妄称为“中国病毒”。而现场有《华盛顿邮报》的摄影记者发现，特朗普的演讲稿上明明标注的是“新冠”病毒，但却被划掉，写上了“中国”病毒。"
keyws = textrank(text,topK=10, withWeight=True, withFlag=True)

for key in keyws:
    print(key)

# output
(pair('新冠', 'n'), 1.0)
(pair('肺炎', 'n'), 0.6579707898718974)
(pair('病毒', 'n'), 0.6501736817899595)
(pair('工作', 'vn'), 0.5709705184797785)
(pair('疫情', 'n'), 0.5501469909757243)
(pair('准备', 'v'), 0.45307574360391695)
(pair('美国', 'ns'), 0.4488577408225489)
(pair('中国', 'ns'), 0.42552723688506267)
(pair('测验', 'vn'), 0.42274922572055534)
(pair('发现', 'v'), 0.4027839511084109)
```

pair为一对的意思

Hanlp**词性列表**
a 形容词
ad 副形词
ag 形容词性语素
al 形容词性惯用语
an 名形词
b 区别词
begin 仅用于始##始
bg 区别语素
bl 区别词性惯用语
c 连词
cc 并列连词
d 副词
dg 辄,俱,复之类的副词
dl 连语
e 叹词
end 仅用于终##终
f 方位词
g 学术词汇
gb 生物相关词汇
gbc 生物类别
gc 化学相关词汇
gg 地理地质相关词汇
gi 计算机相关词汇
gm 数学相关词汇
gp 物理相关词汇
h 前缀
i 成语
j 简称略语
k 后缀
l 习用语

Hanlp**词性列表**
m 数词
mg 数语素
Mg 甲乙丙丁之类的数词
mq 数量词
n 名词
nb 生物名
nba 动物名
nbc 动物纲目
nbp 植物名
nf 食品，比如“薯片”
ng 名词性语素
nh 医药疾病等健康相关名词
nhd 疾病
nhm 药品
ni 机构相关（不是独立机构名）
nic 下属机构
nis 机构后缀
nit 教育相关机构
nl 名词性惯用语
nm 物品名
nmc 化学品名
nn 工作相关名词
nnd 职业
nnt 职务职称
nr 人名
nr1 复姓
nr2 蒙古姓名
nrf 音译人名
nrj 日语人名
ns 地名

Hanlp**词性列表**
nsf 音译地名
nt 机构团体名
ntc 公司名
ntcb 银行
ntcf 工厂
ntch 酒店宾馆
nth 医院
nto 政府机构
nts 中小学
ntu 大学
nx 字母专名
nz 其他专名
o 拟声词
p 介词
pba 介词“把”
pbei 介词“被”
q 量词
qg 量词语素
qt 时量词
qv 动量词
r 代词
rg 代词性语素
Rg 古汉语代词性语素
rr 人称代词
ry 疑问代词
rys 处所疑问代词
ryt 时间疑问代词
ryv 谓词性疑问代词
rz 指示代词
rzs 处所指示代词

Hanlp**词性列表**
rzt 时间指示代词
rzv 谓词性指示代词
s 处所词
t 时间词
tg 时间词性语素
u 助词
ud 助词
ude1 的 底
ude2 地
ude3 得
udeng 等 等等 云云
udh 的话
ug 过
uguo 过
uj 助词
ul 连词
ule 了 喽
ulian 连 （“连小学生都会”）
uls 来讲 来说 而言 说来
usuo 所
uv 连词
uyy 一样 一般 似的 般
uz 着
uzhe 着
uzhi 之
v 动词
vd 副动词
vf 趋向动词
vg 动词性语素
vi 不及物动词（内动词）

Hanlp**词性列表**
vl 动词性惯用语
vn 名动词
vshi 动词“是”
vx 形式动词
vyou 动词“有”
w 标点符号
wb 百分号千分号，全角：% ‰ 半角：%
wd 逗号，全角：， 半角：,
wf 分号，全角：； 半角：;
wh 单位符号，全角：￥ $ £ ° ℃ 半角：$
wj 句号，全角：。
wky 右括号，全角：） 〕 ］ ｝ 》 】 』 〉 半角：)]{>
wkz 左括号，全角：（ 〔 ［ ｛ 《 【 『 〈 半角：([{<
wm 冒号，全角：： 半角：:
wn 顿号，全角：、
wp 破折号，全角：—— - - ——- 半角：— —-
ws 省略号，全角：…… …
wt 叹号，全角：！
ww 问号，全角：？
wyy 右引号，全角：”’』
wyz 左引号，全角：“‘『
x 字符串
xu 网址URL
xx 非语素字
y 语气词(delete yg)
yg 语气语素
z 状态词
zg 状态词