红色标注的语句，为重点。
蓝色下划线标注的语句，说明给出了参考阅读链接，可依兴趣阅读。
紫色加粗，表示参看附件。

写在Lecture8之前

本次Python分享即将进入收尾阶段，接下来三讲的主题是“深度学习与自然语言处理”，我计划Lecture8用来介绍一些基本的文本处理知识，Lecture9主要介绍深度学习的环境搭建和应用，Lecture10介绍深度学习下的自然语言处理，以及对全部内容做一个总结。

本节内容按以下方式组织：

前几天从师兄那看到了华泰的一张slide，是介绍Python的文字处理功能在金融行业日常工作时的应用场景的：

参照这张slide，我们模拟了一个日常使用Python工作的场景，将这些需求串起来，通过实例让大家更好地感受Python对于日常办公的助力。

涉及到的环节如下：1.根据文件的创建时间将文件改名、重新整理。2.从pdf文件中读取信息。3.从文本信息中自动提取自己需要的部分。4.用代码进行中文分词。5.从文本中自动提取摘要和关键词。6.使用百度api调用NLP功能。

（鉴于用Python生成词云效果还不如在线服务，而且要解决中文乱码问题，所以不推荐）

1. 文件整理

——*os*模块

Python中，涉及到文件操作和路径操作的功能，主要由os模块完成，这是一个系统自带的模块。

os模块可以实现返回绝对路径、列出路径下所有文件、查询文件修改时间等功能，具体可以参照：https://www.runoob.com/python3/python3-os-file-methods.html

我们更常使用的是它的path子模块，参照：

https://www.runoob.com/python3/python3-os-path.html

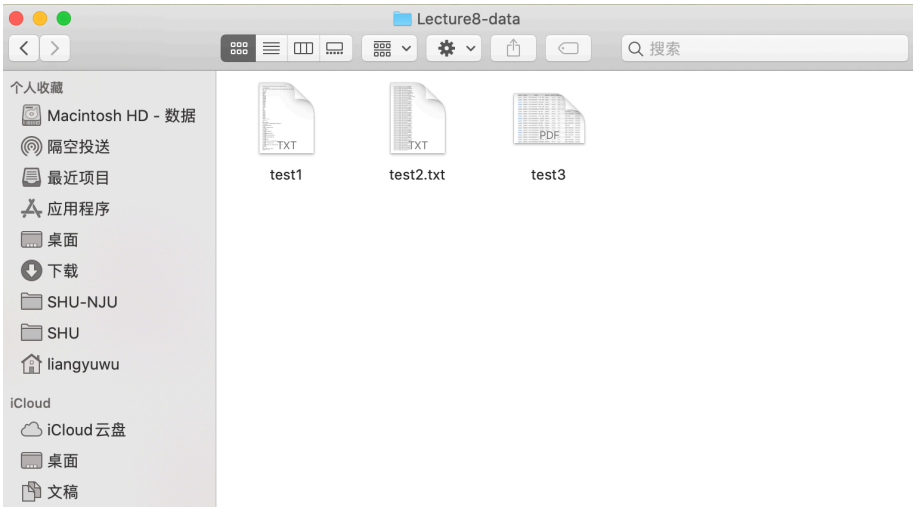
两者中部分常用的功能如下（更多还请参照菜鸟教程或官方文档）：

函数名	功能
os.read(fd,d)	从文件fd中读取n个字节
os.remove(path)	删除路径为path的文件（非文件夹）
os.removedirs(path)	递归删除路径为path的文件夹
os.rename(src,dst)	重命名，从src到dst
os.getcwd( )	获取当前路径
os.listdir(path)	列出当前文件夹下的文件名
os.path.abspath(path)	返回path的绝对路径
os.path.getctime(path)	返回文件创建时间
os.path.getmtime(path)	返回文件修改时间

——相对路径与绝对路径

在使用路径来查找、创建、存储文件时，一定要注意绝对路径和相对路径的区别。简单来说，绝对路径就是你在人口普查时，需要上报的居住地址，是尽可能详细的、从最高一级行政区域开始（中国江苏省南京市）；而相对路径是你在像其他人指路时，从你当前所在位置给出的地址，是以当前路径为参考系的（从这里往前直走再左转就到了）。

在相对路径中，我们用./表示当前路径，用../表示上级路径，它们之间可以叠加，如../../表示两层上级路径。



如图，当前的工作路径是Lecture8-data，它的绝对路径是/Users/liangyuwu/Documents/NJU/Python/Lecture8-data，如果我想访问当前路径下的test1.txt文件，绝对路径写法是/Users/

liangyuwu/Documents/NJU/Python/Lecture8-data，相对路径写法是./test1.txt。如果用访问当前路径的父文件夹的父文件夹下的Paper文件夹，绝对路径写法是/Users/liangyuwu/Documents/NJU/Paper，相对路径写法是../../Paper。

关于绝对路径和相对路径，可参考：http://c.biancheng.net/view/5862.html （Linux版，正好方便大家进一步熟悉Linux的操作逻辑）。

Ps：为了使你的代码复用性高，可以在不同的机器上顺利运行，请尽量使用相对路径。毕竟，其他电脑是看不懂那些只有在你的电脑上能识别的绝对路径的。

——使用Python批量修改文件名称

见Lecture8-附件。

2. 读取pdf文件的信息

Pdf转文字的需求一直都存在，以至于网上涌现了很多的收费服务，但其实很多中小型收费服务的后台都是用Python实现的，技术并不高明。

不过，即便Python生态下有大量开发者提供了免费的pdf识别模块，对pdf文件的读取和转化文字效果也并不是完美的。在这一领域，可能某些大型商业化公司如Adobe、PDFExpert推出的付费软件的识别效果更佳准确。Python的优势可能在于大批量的自动化识别和部署。

常见的用于读取pdf的模块包括pdfplumber、pdfminer、pyPDF2等，它们都有各自的特色和缺陷，主要体现在对表格内容的识别上。

详细的对比可见：https://blog.csdn.net/Asher117/article/details/89203780

我们的实例以pdfplumber为例，见Lecture8-附件。

3.使用正则表达式提取信息

在前两步中，我们成功修改、批量读取了各类文件，但很多时候，我们需要的仅仅是文件中的一部分特定信息，如证券代码、手机号、邮箱等。这就需要我们编写代码从海量的字符中，提取我们需要的部分。

——正则表达式

匹配字符串，需要正则表达式的出场。

正则表达式是一种字符串匹配的模式，它可以将我们对字符串格式的要求以式子的形式存储下来，让程序去识别，从而实现查找一个字符串是否含有某个子串等功能。

正则表达式的相关知识可参考：https://www.runoob.com/regexp/regexp-tutorial.html

可以用以下五条总结来快速入门正则表达式：

第一，正则表达式用模式字符串来表达我们想要匹配的规则，模式字符串是一个普通的字符串，但是是用来代表一类符合其规则的字符串。可以类比我们常说的AABB、AABC式词语。

第二， 模式字符串中，非转义字符代表其本身，这些字符连接起来就变成模式字符串。如’abc’就是用来匹配字符串中的’abc’的。我们也可以用中括号表示匹配任意一个处在中括号中的字符，如[a-z]表示匹配任意一个小写字母，[0-9]表示匹配任意一个数字，[123]表示匹配任意一个1或者2或者3。

第三，模式字符串用转义字符来进行特殊的匹配，转义字符大多为在原字符上加一个反斜杠\，常见的有.匹配任意一个字符，\s匹配任意一个空白字符，\w匹配任意一个字母、数字、下划线字符，\d匹配任意一个数字字符。往往不同大小写之间的转义字符意思是相反的，如\S匹配任意一个非空白字符，\W匹配任意一个非字母、数字、下划线字符，\D匹配任意一个非数字字符。

第四，模式字符串有方便快捷的重复方法，只需将如下符号置于想重复的部分后面，*表示重复任意次，+表示重复一次以上，?表示重复0或1次，{x}表示重复x次。

第五，正则表达式也可以用来做前瞻和后顾的匹配，比如匹配前面是xxx，或者后面是xxx的位置。

（更多详细语法，见Lecture8-附件-正则表达式常用语法）

——*re*模块

Python中，涉及到正则表达式的功能，主要由re模块完成，这是一个系统自带的模块。

re模块的使用可参考：https://www.runoob.com/python3/python3-reg-expressions.html

下面看一个实例：

```
import re

text = '''歪你打错了！我不是wly！重要的事情说31237938123789遍，我的手机号是13512346789，不是15600998765，
也不是110或119或者19023456789，我的手机号是15600998765，别再打给我啦！'''

#beta 1
regex1 = re.compile(r'\d{11}') #re.compile函数构建一个正则表达式对象
result1 = regex1.findall(text) #调用该对象的.findall()方法，在传入的参数text中找到符合正则表达式语法的部分
print(result1)

#beta 2
regex2 = re.compile(r'(?<=\D)1[3578]\d{9}(?=\D)') #同上，注意这里正则表达式前瞻和后顾的写法，以及增加了对于手机号开头两位数的细化
result2 = regex2.findall(text)
print(result2)
```

```
['31237938123', '13512346789', '15600998765', '19023456789', '15600998765']
['13512346789', '15600998765', '15600998765']
```

注意到两点：第一，我们需要使用re.compile函数将传入的模式字符串，转换成一个正则表达式对象，后续的操作都是在调用这个表达式对象的方法，如使用findall方法，将符合条件的子串整合成一个列表List返回。第二，留意到(?<=\D)(?=\D)这一对式子起到的“瞻前顾后”效果，它们共同指定了匹配的字符串前后均非数字，这样就成功避免了实例文本中3123792812379这一超长字符串带来的干扰效果。

——用*re*模块提取信息

见Lecture8-附件。

4. 中文分词

这一部分是后一部分“摘要和关键词”的技术基础，因此在此简单介绍。

中文分词是自然语言处理（NLP）的一项基本技术，也是几乎所有NLP的基础。中文分词，顾名思义，就是/将/一段/长长/的/中文/字符串/拆成/一个个/词组/便于/计算机/理解。由于中文具有很强的歧义性，因此相对英文来说更为困难。目前主要有暴力字符匹配、句法分析、基于统计的学习方法这三种思路，在实际操作中，业界主要是采用三种思路混合的方式。

当然，我们不需要关心业界的技术逻辑，Python生态系统中，jieba分词工具已经很好地将中文分词技术打包起来，供我们方便地调用。

在分词过程中，我们只需要设置好停用词（即一些无意义的词，如的、得）、用户词典（即用户不希望被切开的专有名词，比如工程管理学院），指定好分词方法，就可以进行分词了。

作为补充，我还推荐一个我自己比较喜欢的分词工具：清华大学自然语言处理与社会人文计算实验室出品的THULAC，它的词性标注效果令人印象深刻。

无论是jieba还是THULAC，官方都提供了翔实的入门指引：

Jieba：https://github.com/fxsjy/jieba

THULAC：http://thulac.thunlp.org

5.提取摘要和关键词

要在短时间内阅读几百篇新闻，并进一步缩小阅读范围？使用Python进行批量摘要提取可以方便我们筛选感兴趣的文章。

***——TextRank*算法**

我们使用TextRank算法来判断一篇文章中，哪些词语、句子是重要的，从而实现提取摘要和关键词。

TextRank脱胎于Google发家并引以为傲的PageRank算法（这也是现代搜索引擎的算法基础），其采用了投票的方式，根据“被重要网页引用的链接一定也很重要”的原则来对网页重要性进行排序。

TextRank的知识可以阅读：https://zhuanlan.zhihu.com/p/55270310

***——TextRank4ZH*模块**

国人开发了TextRank4ZH模块来实现中文文档的TextRank算法。TextRank4ZH模块以jieba、numpy和网络数据模块networkx为基础。

模块的使用比较简单，实例见Lecture8-附件

6.使用api调用NLP功能

我们可以使用自然语言处理来实现众多功能：文本相似度、评论观点抽取、情感分析、文章分类、地址识别。

百度开放了其接口，为我们提供方便的调用，使用百度的接口，需要注册好百度的智能云服务，设置好应用id和密钥，并且安装好baidu-aip模块。

应用id和密钥参照如下设置：https://www.cnblogs.com/zlc364624/p/12482427.html

在配置好环境后，我们通过这样的方式调用百度的接口：

```
from aip import AipNlp

APP_ID = '18102862'
API_KEY = 'igU7dumhhWws35yIMUE6wGRL'
SECRET_KEY = 'hE9QieKEA3nYUrGIbKVbIdrmEZGsUGgS'

client = AipNlp(APP_ID, API_KEY, SECRET_KEY)

text1 = "工程管理学院"

text2 = "信息管理学院"

""" 调用短文本相似度 """
client.simnet(text1, text2);

""" 带参数调用短文本相似度 """
client.simnet(text1, text2)
```

（大家可以将我的APP_ID用作测试，但不建议正式工作里使用，建议自己注册）

可以看到，在使用client对象完成链接后，所有的操作都是基于client对象的。

具体见Lecture8-附件