第3章

Python下的实际应用

Python是一种代表简单主义思想的语言，有“胶水语言”之称。目前，Python的应用层面也十分广泛，主要包括数据库的连接、系统编程、网络爬虫、人工智能、Web开发、系统运维、大数据、云计算、图形界面等。在当前环境下，Python重点应用于机器学习和人工智能方面。

3.1 Python连接MySQL数据库

视频讲解

MySQL数据库是企业最常用的数据库之一，而MySQL数据库最常用的功能是select查询。数据分析人员希望可以用Python直接连接MySQL数据库，然后读取数据到pandas框架中。

首先安装好MySQL数据库，安装步骤详见附录B。然后安装pymysql，安装语句为：pip install pymysql，安装好这个包以后，在Jupyter Notebook中使用以下代码：

```
import pandas as pd
import pymysql
conn = pymysql.connect(
    host = '主机 IP',
```

```
    user = '用户名',
    password = '密码',
    db = '数据库名',
    port = 端口,
    charset = 'utf8')
table = "select...from..."
```

上面的 table 就是连接 mysql 后查询到的表。然后用下面代码,就可以直接把数据读取到 pandas 框架中了。数据读取到 pandas 框架后,就可以使用 pandas 的常见功能对数据进行处理,比如填补缺失值、删除数据等。

```
data = pd.read_sql(table, conn)
```

3.2 Python 连接 MongoDB 数据库

视频讲解

MongoDB 数据库也是企业常用的数据库之一。MongoDB 数据库是一个介于关系数据库和非关系数据库之间的产品,是非关系数据库中功能最丰富、最像关系数据库的一种数据库。

相对于 MySQL 数据库,MongoDB 数据库具有独特的优势。同样地,Python 也可以直接连接 MongoDB 数据库,并把数据读取到 pandas 框架中。安装步骤详见附录 C。

首先安装 pymongo,安装语句为:pip install pymongo。安装步骤详见附录 F。安装好 pymongo 后,使用方法如下:

```
import pandas as pd
from pymongo import MongoClient
client = MongoClient('主机 IP', port = 端口)
db = client.数据库名
db.authenticate('用户名', '密码')
table = db.表名
```

与 MySQL 数据库一样,上面的 table 就是连接 MongoDB 后查询到的表,然后使用下面代码,就可以直接把数据读取到 pandas 框架了,其中,find 括号里可以加入 MongoDB 数据库查询语句。

```
data = pd.DataFrame(list(table.find()))
```

3.3 结巴分词和词云图

视频讲解

Python可以处理文本数据，常见的有NLTK自然语言处理工具包、LSTM长短期记忆网络，但这些工具包基本上都用于处理英文。这里介绍下简单且常用的中文语言处理工具包jieba。

下面将以Mac系统为例讲解jieba和wordcloud的安装方法，安装语句为：pip install jieba，pip install wordcloud。步骤参考附录F第三方包安装步骤。

本节以网络下载的小说《天龙八部》为例，讲解jieba和word cloud的用法。

```
import pandas as pd
import os
os.chdir('../data')
character = open('天龙八部人物表.txt', encoding = 'utf - 8').read()
'''添加词性'''
import re
data = re.split(r'\s + | : | ,', character)
data = pd.DataFrame(data, columns = ['姓名'])
data['词性'] = 'nr'
data.to_excel('天龙八部人物分词.xlsx', index = False, header = None)
'''添加自定义词库'''
import jieba
jieba.enable_parallel(4)
jieba.load_userdict('天龙八部人物词典.txt')
'''加载停用词库'''
stopwords = [line.rstrip() for line in open('停用词表.txt', encoding = 'utf - 8')]
'''jieba 分词'''
def seg_sentence(sentence):
    sentence_seged = jieba.cut(sentence.strip())
    outstr = ''
    for word in sentence_seged:
        if word not in stopwords:
            if word != '\t':
                outstr += word
                outstr += ''
    return outstr
inputs = open('天龙八部.txt', 'r', encoding = 'GB18030')
outputs = open('天龙八部分词.txt', 'w', encoding = 'utf - 8')
```

```
for line in inputs:
    line_seg = seg_sentence(line)
    outputs.write(line_seg + '\n')
outputs.close()
inputs.close()
```

分词过后,就可以进行文本分析了。先加载分词后的 txt 文本:

```
text = open('天龙八部分词.txt', encoding = 'utf - 8').read()
```

然后指定词性,提取关键词,并打印出由 TF-IDF 方法计算出的权重最大的前 10 人物,效果如图 3-1 所示。

```
'''指定词性, 提取关键词: TF-IDF'''
import jieba.analyse

n=100           #指定关键词数量

result=jieba.analyse.extract_tags(text, topK=n, withWeight=True, allowPOS=('nr',))

result[:10]     #打印前10个最重要的人物
```

```
[('段誉', 0.5891293630142107),
 ('萧峰', 0.4638848540025745),
 ('乔峰', 0.3369411093846939),
 ('慕容复', 0.2771830168385107),
 ('王语嫣', 0.26712617741403344),
 ('武功', 0.23669144784504287),
 ('段正淳', 0.22647654172059356),
 ('阿朱', 0.21979838728524273),
 ('木婉清', 0.21805626004123815),
 ('鸠摩智', 0.1742127244004566)]
```

图 3-1　TF-IDF 计算出的权重最大的前 10 人

```
'''词云关键词'''
keywords = dict()
for i in result:
    keywords[i[0]] = i[1]
'''设置词云属性'''
from PIL import Image
import numpy as np
from wordcloud import WordCloud, ImageColorGenerator
import matplotlib.pyplot as plt
image = Image.open('图片.jpeg')
graph = np.array(image)
wc = WordCloud(font_path = '/Library/Fonts/Songti.ttc',    #设置字体
               background_color = 'White',                 #设置背景颜色
               max_words = n,                              #设置词云显示的最大词数
               mask = graph)                               #设置背景样式
```

最后效果如图 3-2 所示。

```
In [41]: '''生成词云'''
wc.generate_from_frequencies(keywords)
plt.imshow(wc)
image_color = ImageColorGenerator(graph)
plt.imshow(wc.recolor(color_func=image_color))
plt.axis("off")
plt.show()
wc.to_file('词云.jpg')
```

图 3-2　词云生成图

3.4　简单社交网络

视频讲解

目前，社交网络是大数据时代的热门课题之一，正式的社交网络图谱一般用 Neo4j 来做，Neo4j 的入门与应用将在本书第 8 章中进行详细介绍。本章将介绍 Python 语言中的简单社交网络。

```
import pandas as pd
data1 = pd.read_excel('牛魔王关系图.xlsx')
data2 = pd.read_excel('孙悟空关系图.xlsx')
data3 = pd.read_excel('唐僧关系图.xlsx')
data4 = pd.read_excel('猪八戒关系图.xlsx')
data = pd.concat([data1, data2, data3, data4])
```

简单社交网络数据展示图，如图 3-3 所示。

Source 是源节点，Target 是目标节点，Weight 是关系权重。

在社交网络中有 4 个重要的定义。

(1) 度中心性：一个节点在网络中的连接数。

```
In [5]: data.head()
```

Out[5]:

	Source	Target	Weight
0	牛魔王	铁扇公主	7
1	牛魔王	红孩儿	7
2	牛魔王	孙悟空	5
3	牛魔王	玉面狐狸	6
4	铁扇公主	红孩儿	8

图 3-3　简单社交网络数据展示图

(2) 接近中心性：识别集群内部被高度连接的节点。

(3) 中介中心性：识别连接不同集群的节点。

(4) 网页排名：衡量节点活跃度，节点越活跃，pangrank 值越大。

```
import networkx as nx
graph = nx.from_pandas_dataframe(data, 'Source', 'Target', edge_attr=['Weight'])
```

度中心性、接近中心性、中介中心性、网页排名分别如图 3-4、图 3-5、图 3-6 和图 3-7 所示。

```
'''度中心性：一个节点在网络中的连接数'''
degree_df = pd.DataFrame(columns=['度中心性'])
degree_df['度中心性'] = pd.Series(nx.degree_centrality(graph))
degree_df.sort_values('度中心性', ascending=False)[:10].plot(kind='barh')
plt.show()
```

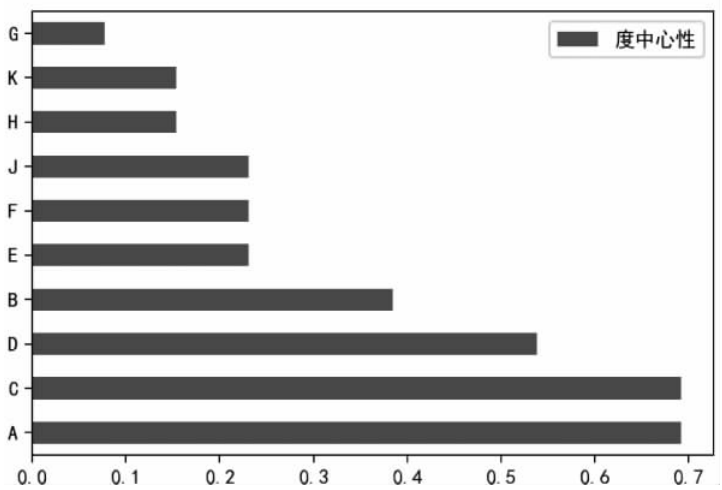


图 3-4　度中心性展示图

```
'''接近中心性：具有高接近中心性的节点通常在集群之间被高度连接'''
closeness_df = pd.DataFrame(columns=['接近中心性'])
closeness_df['接近中心性'] = pd.Series(nx.closeness_centrality(graph))
closeness_df.sort_values('接近中心性', ascending=False)[:10].plot(kind='barh')
plt.show()
```

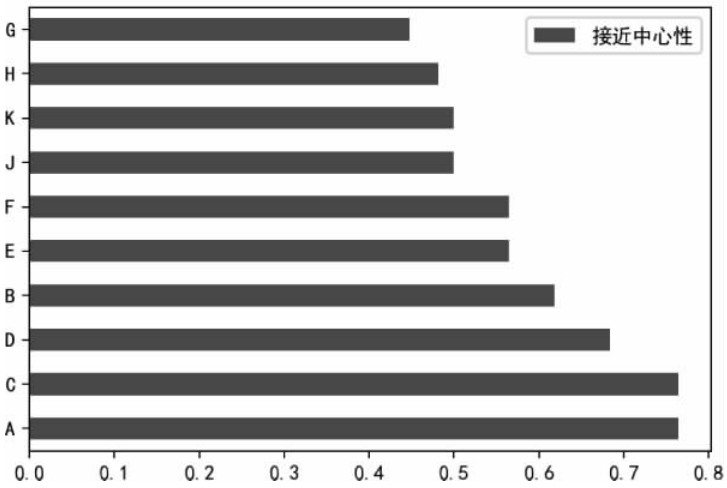


图 3-5　接近中心性展示图

```
'''中介中心性：识别连接不同集群的节点'''
betweenness_df = pd.DataFrame(columns=['中介中心性'])
betweenness_df['中介中心性'] = pd.Series(nx.betweenness_centrality(graph))
betweenness_df.fillna(0, inplace=True)
betweenness_df.sort_values('中介中心性', ascending=False)[:10].plot(kind='barh')
plt.show()
```

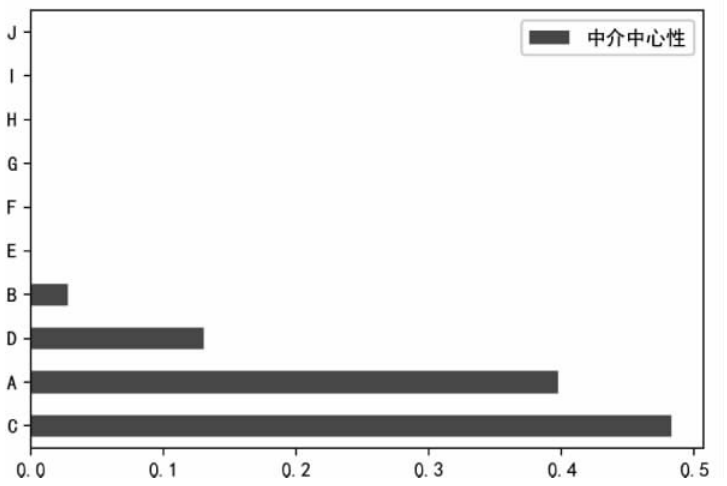


图 3-6　中介中心性展示图

```
'''网页排名：找出活跃用户'''
pagerank_df = pd.DataFrame(columns=['网页排名'])
pagerank_df['网页排名'] = pd.Series(nx.pagerank(graph))
pagerank_df.sort_values('网页排名', ascending=False)[:10].plot(kind='barh')
plt.show()
```

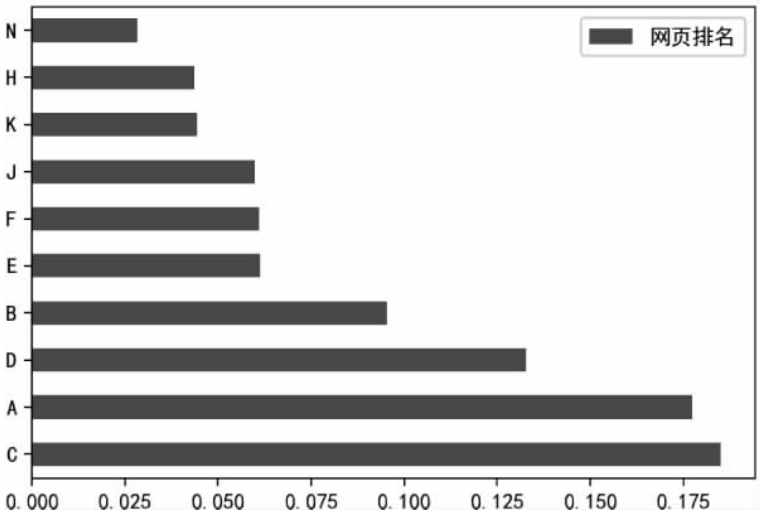


图 3-7　网页排名展示图

社交网络可视化代码如下：

```
plt.figure(figsize=(20, 10))
node_color = [graph.degree(v) for v in graph]
#  节点颜色由节点的度决定
node_size = [5000 * nx.degree_centrality(graph)[v] for v in graph]
#  节点的大小由 degree centrality 决定
edge_width = [0.2 * graph[u][v]['Weight'] for u, v in graph.edges()]
#  边的宽度由权重决定
pos = nx.spring_layout(graph)
nx.draw_networkx(graph, pos, node_size=node_size, node_color=node_color, alpha=0.7,
with_labels=False, width=edge_width)
top = degree_df.sort_values('度中心性', ascending=False)[: 10]
#  最重要的 top10 人物
top = top.index.values.tolist()
labels = {role: role for role in top}
#  创建 label,不同的度量方法.标签不一样
nx.draw_networkx_labels(graph, pos, labels=labels, font_size=20)
#  添加 label
plt.show()
nx.write_gexf(graph, '度中心性.gexf')
```

社交网络可视化结果，如图 3-8 所示。

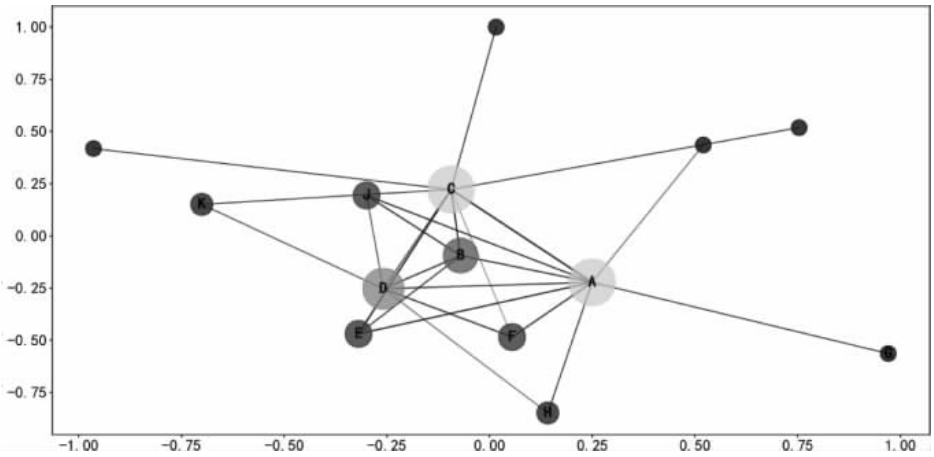


图 3-8　社交网络可视化结果展示图

最小社区探测代码如下：

```
k = 3
klist = list(nx.k_clique_communities(graph,k))
print ("生成的社区数: %d" % len(klist))
plt.figure(figsize=(20, 10))
nx.draw_networkx(graph, pos = pos, nodelist = klist[0], node_size=node_size, node_color
= 'r', alpha=0.7, with_labels=False, width=edge_width)
nx.draw_networkx(graph, pos = pos, nodelist = klist[1], node_size=node_size, node_color
= 'y', alpha=0.7, with_labels=False, width=edge_width)
nx.draw_networkx_labels(graph, pos, labels=labels, font_size=20)
plt.show()
```

最小社区探测可视化结果，如图 3-9 所示。

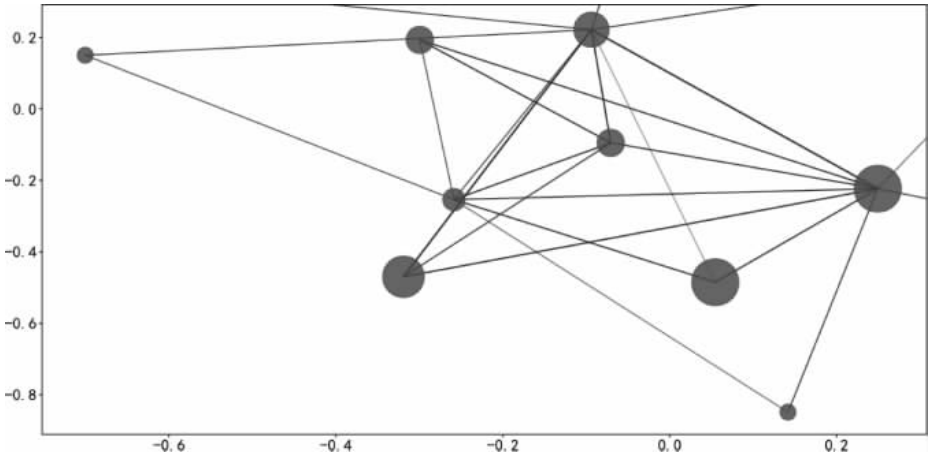


图 3-9　最小社区探测可视化展示图

上述代码指定 $k=3$，即一个社区至少有 3 个节点两两相连，而这里生成了两个社区。在金融领域，社区探测法常用于寻找欺诈团体。

3.5 JSON 解析

视频讲解

在进行数据分析时，从数据库中导出的数据，可能是 JSON 格式的。那么 JSON 是什么呢？JSON 是 JavaScript Object Notation 的缩写，是一种轻量级的数据交换格式，或者说是一个像字典一样的字符串。下面以读取 1 个 JSON 文件为例，向大家进行展示。

```
import json
with open('food.json', 'r') as f:
    data = json.load(f)
```

预览结果如图 3-10 所示。

```
In [34]: data[0]

Out[34]: {'description': 'Cheese, caraway',
          'group': 'Dairy and Egg Products',
          'id': 1008,
          'manufacturer': '',
          'nutrients': [{'description': 'Protein',
            'group': 'Composition',
            'units': 'g',
            'value': 25.18},
           {'description': 'Total lipid (fat)',
            'group': 'Composition',
            'units': 'g',
            'value': 29.2},
           {'description': 'Carbohydrate, by differenc
            'group': 'Composition',
            'units': 'g',
            'value': 3.06},
           {'description': 'Ash', 'group': 'Other', 'u
           {'description': 'Energy',
            'group': 'Energy',
```

图 3-10　food.json 数据预览展示图

以上就是文件 food.json 里面的内容。通常，理想的格式是像 pandas 那样标准的格式，该怎么处理呢？

仔细观察上面的 JSON 文件，可以发现有 5 个主键：description、group、id、manufacturer 和 nutrients。其中前 4 个键的取值是字符串，但 nutrients 的取值是一个列表，列表中包含的是另一个 JSON 格式的数据。

基于上述规律，先处理单独的 4 个主键（description、group、id 和 manufacturer），再处理主键 nutrients 的内容。

```
keys = ['id','description','group']
keys_df = pd.DataFrame(data, columns = keys)
nutrients = []
for i in data:
    temp = pd.DataFrame(i['nutrients'])
    temp['id'] = i['id']
    nutrients.append(temp)
nutrients = pd.concat(nutrients, ignore_index = True)
```

最终结果如图 3-11 所示。

```
In [36]: nutrients.head()
```

Out[36]:

	description	group	units	value	id
0	Protein	Composition	g	25.18	1008
1	Total lipid (fat)	Composition	g	29.20	1008
2	Carbohydrate, by difference	Composition	g	3.06	1008
3	Ash	Other	g	3.28	1008
4	Energy	Energy	kcal	376.00	1008

图 3-11 解析 food.json 中的 nutrients 数据

下面处理略微复杂一些的 JSON。

```
with open('report.json', 'r') as f:
    data = json.load(f)
```

【注意】

这个 JSON 文件涉及敏感信息，所以本书仅作代码示例，不提供文件示例。

预览结果如图 3-12 所示。

```
In [40]: data[0]

Out[40]: {'create_time': 1504088427000,
          'id': 1233,
          'modify_time': None,
          'mongo_id': None,
          'report_data': '{"trip_analysis":[{"called_cnt":"620","talk_s
         led_seconds":"58635","receive_cnt":"0","date_distribution":["2
         -03"],"detail":[{"called_cnt":"151","talk_seconds":"29980","ta
         1","receive_cnt":"0","month":"2017-08","call_cnt":"204","unkno
         {"called_cnt":90,"talk_seconds":14568,"talk_cnt":186,"msg_cnt'
         7-07","call_cnt":96,"unknown_cnt":0,"send_cnt":0,"call_seconds
```

图 3-12 report.json 数据预览展示图

下面展示解析 report_data 里的 trip_analysis 字段的代码示例。

```
trip_analysis = []
for i in data:
    str1 = i['report_data']
    dict1 = json.loads(str1)
    temp = pd.DataFrame(dict1['trip_analysis'])
    temp['user_id'] = i['user_id']
    trip_analysis.append(temp)
trip_analysis = pd.concat(trip_analysis, ignore_index = True)
```

最终结果如图 3-13 所示。

```
In [43]: trip_analysis.head(1)
```

Out[43]:

	call_cnt	call_seconds	called_cnt	called_seconds	date_distribution
0	847	66569	620	58635	[2017-08, 2017-07, 2017-06, 2017-05, 2017-04, ...

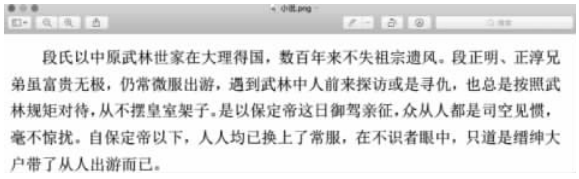
图 3-13　解析 report.json 中的 trip_analysis 数据

3.6　OCR 文字识别

视频讲解

先给大家看一张带有文字内容的图片，如图 3-14 所示。

将图片识别成文字，如图 3-15 所示。



段氏以中原武林世家在大理得国，数百年来不失祖宗遗风。段正明、正淳兄弟虽富贵无极，仍常微服出游，遇到武林中人前来探访或是寻仇，也总是按照武林规矩对待，从不摆皇室架子。是以保定帝这日御驾亲征，众从人都是司空见惯，毫不惊扰。自保定帝以下，人人均已换上了常服，在不识者眼中，只道是缙绅大户带了从人出游而已。

图 3-14　小说.png 图片展示图

```
zhaikundeiMac:~ zhaikun$ python ocr.py
请输入文件名：小说.png
已收到，正在处理，请稍后...
处理完成
zhaikundeiMac:~ zhaikun$
```

图 3-15　图片识别程序运行展示图

下面是转化后的效果，如图 3-16 所示。

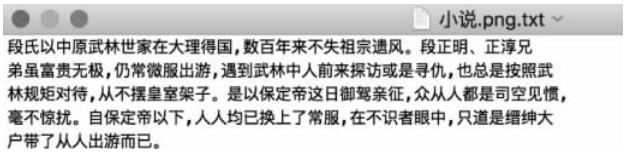


图 3-16　图片识别为文字之后的 txt 文档展示图

是不是很酷！这是怎么做的呢？其实操作很简单，写一段简单的代码，调用百度 API 即可。

先打开网址：https://login.bce.baidu.com/，登录百度账号，如图 3-17 所示。

图 3-17　登录百度云页面展示图

然后单击“创建应用”按钮，就能获取 AppID、API Key、Secret Key，如图 3-18 和图 3-19 所示。

图 3-18　创建文字识别应用

图 3-19　应用名称及对应的 AppID、API Key、Secret Key 展示图

然后安装 baidu-aip，安装语句为：pip install baidu-aip，安装教程详见附录 F。安装好后，代码如下。

```
from aip import AipOcr
import codecs
import os
os.chdir('../data')
def ocr(path):
    with open(path, 'rb') as f:
        return f.read()
def main():
    file_name = str(input('请输入文件名:'))
    print ('已收到,正在处理,请稍后...')
    app_id = '你自己的 id'
    api_key = '你自己的 api_key '
    secret_key = '你自己的 secret_key '
    client = AipOcr(app_id, api_key, secret_key)
    image = ocr(file_name)
    dict1 = client.general(image)
    with codecs.open(file_name + ".txt", "w", "utf-8") as f:
        for i in dict1['words_result']:
            f. write(str(i['words']) + "\r\n")
    print ('处理完成')
if __name__ == '__main__':
    main()
```

通过运行以上代码就可以成功调用百度的 API,从而进行 OCR 识别了。除了普通的文字识别,还能进行以下特殊文件识别,如图 3-20 所示。

图 3-20　百度 OCR 的其他文件识别展示图

具体介绍的链接地址为：https://cloud.baidu.com/doc/OCR/OCR-API.html。

3.7　pyecharts

视频讲解

ECharts是一个使用JavaScript实现的开源可视化库，可以流畅地运行在PC端和移动设备上，提供直观的、交互丰富的、可高度个性化定制的数据可视化图表，官网链接地址为：http://echarts.baidu.com/。用ECharts生成的图的可视化效果非常棒，为了与Python进行对接，同时方便在Python中直接使用数据生成图，国内开发人员开发出了第三方包：pyecharts。

首先安装pyecharts，安装语句为：pip install pyecharts，安装教程详见附录F。安装好后，打开Jupyter Notebook。pyecharts有交互效果，所以需要用Jupyter Notebook运行以下代码。

先读取数据：

```
import pandas as pd
import os
os.chdir('../data')
data = pd.read_excel('房价与工资.xlsx')
```

数据预览如图3-21所示。

```
In [3]: data.head(3)
```

Out[3]:

	城市	省份	房屋均价	人均工资
0	北京市	北京	63239	9240
1	重庆市	重庆	10351	5962
2	广州市	广东	30894	7409

图3-21　房价与工资.xlsx数据预览展示图

下面指定x轴和y轴：

```
x = list(data['城市'])
y1 = list(data['房屋均价'])
y2 = list(data['人均工资'])
```

先看如何绘制垂直条形图，代码如下，结果如图3-22所示。

```
from pyecharts import Bar
bar = Bar()
```

```
bar.add("房屋均价(元)", x, y1, mark_point = ["max", "min"], mark_line = ["average"], is_
label_show = True)
bar.add("人均工资(元)", x, y2)
bar
```

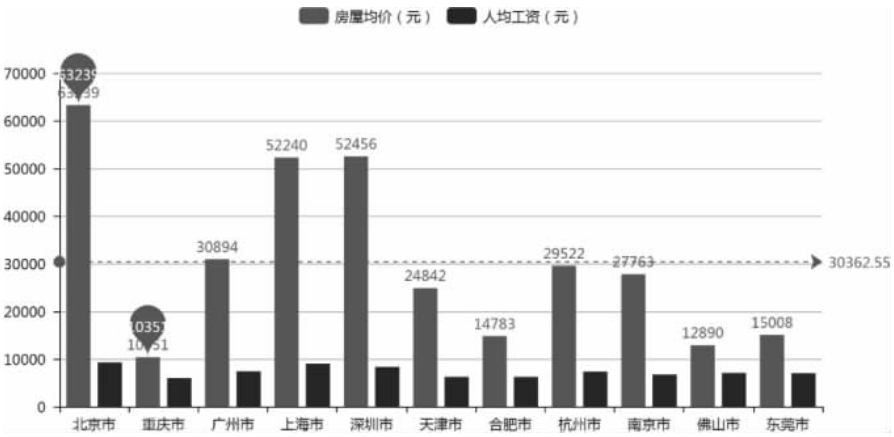


图 3-22　房价与工资.xlsx 数据垂直条形图

图例是可以交互的,单击图 3-22 正上方的“房屋均价(元)”,可得如图 3-23 所示的交互图。

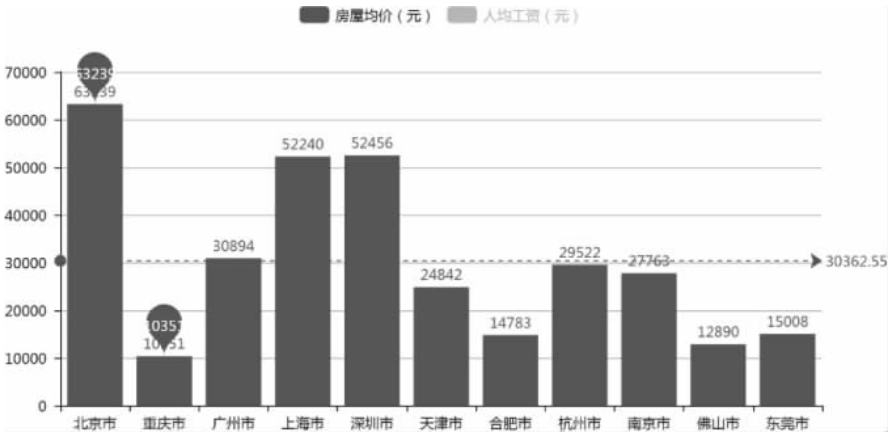


图 3-23　房价与工资.xlsx 数据交互图

再看如何绘制水平条形图,代码如下,结果如图 3-24 所示。

```
from pyecharts import Bar
bar = Bar()
bar.add("房屋均价(元)", x, y1)
bar.add("人均工资(元)", x, y2, is_convert = True)
bar
```

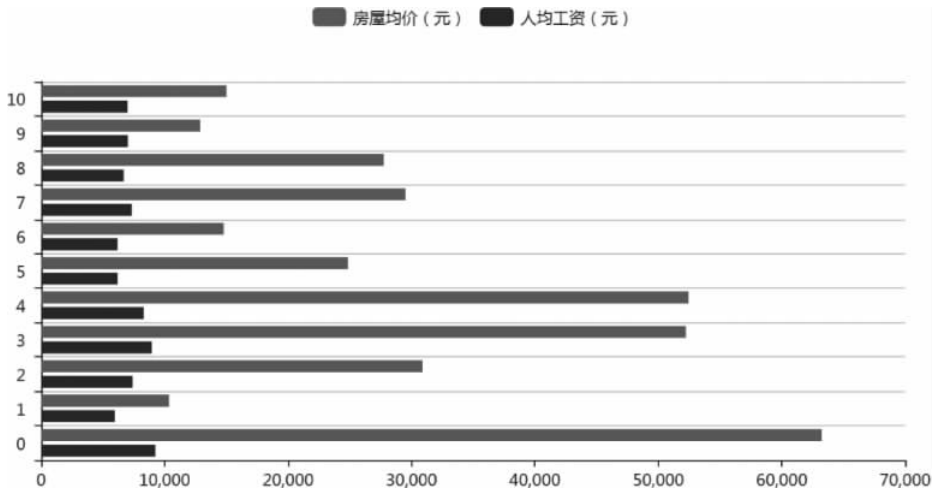


图 3-24　房价与工资.xlsx 数据水平条形图

再看如何绘制直方图，代码如下，结果如图 3-25 所示。

```
from pyecharts import Bar
bar = Bar()
bar.add('房屋均价(元)', x, y1, bar_category_gap = 10, is_label_show = True)
bar
```

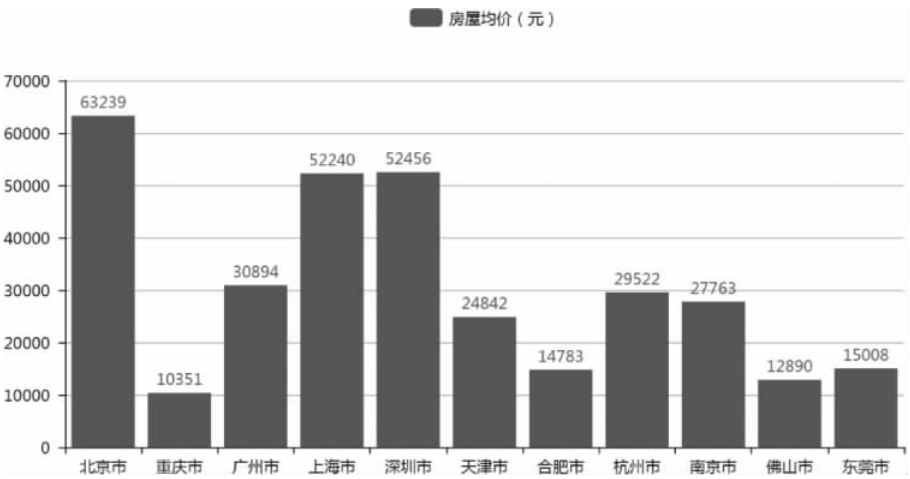


图 3-25　房价与工资.xlsx 的房屋均价直方图

再看如何绘制折线图，代码如下，结果如图 3-26 所示。

```
from pyecharts import Line
line = Line()
```

```
line.add('房屋均价(元)', x, y1, is_label_show = True)
line.add('人均收入(元)', x, y2)
line
```

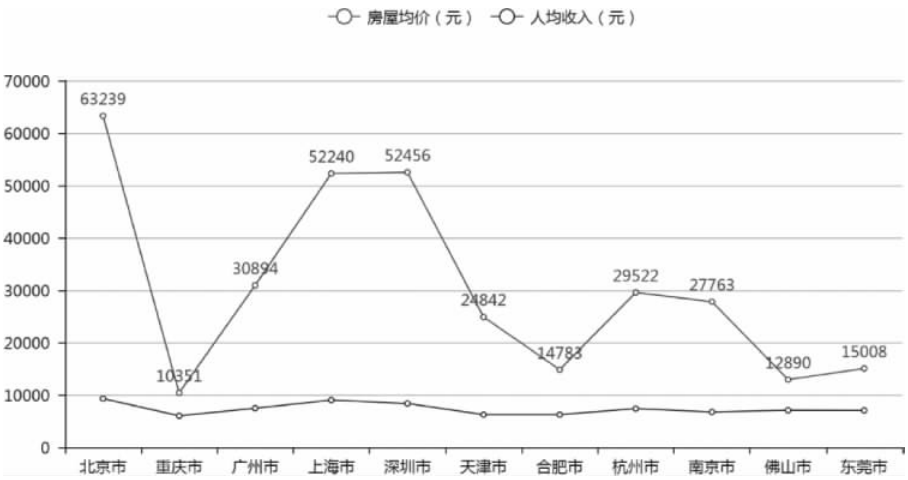


图 3-26　房价与工资.xlsx 的数据折线图

再看如何绘制饼图，代码如下，结果如图 3-27 所示。

```
from pyecharts import Pie
pie = Pie()
pie.add('房屋均价(元)', x, y1, is_label_show = True, legend_orient = 'vertical', legend_pos
= 'left')
pie
```

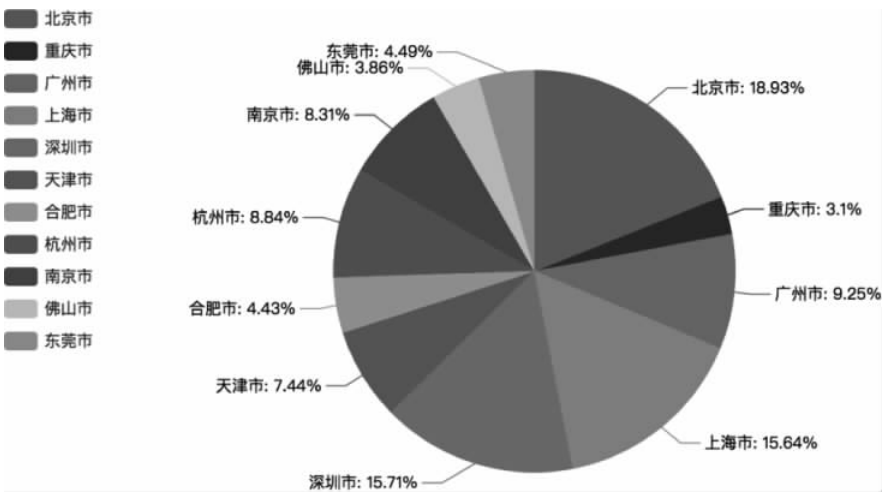


图 3-27　房价与工资.xlsx 的房屋饼图

也可将条形图和折线图绘制在同一个图形里，代码如下，结果如图 3-28 所示。

```
from pyecharts import Bar, Line, Grid
bar = Bar()
bar.add("房屋均价(元)", x, y1, is_label_show = True)
bar.add("人均工资(元)", x, y2)
line = Line()
line.add('房屋均价(元)', x, y1, is_label_show = True)
line.add('人均工资(元)', x, y2)
grid = Grid()
grid.add(bar, grid_bottom = "60 % ")
grid.add(line, grid_top = "60 % ")
grid
```

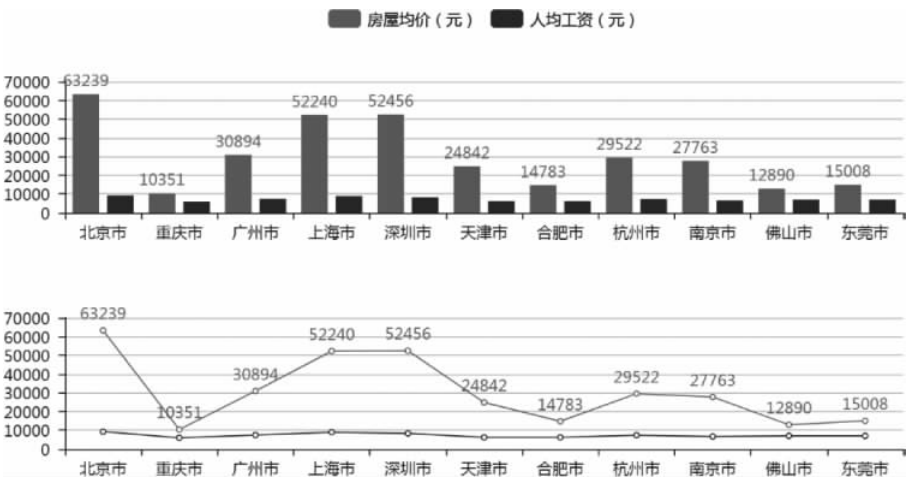


图 3-28　房价与工资.xlsx 的数据条形图和折线图

在条形图上叠加折线图，代码如下，结果如图 3-29 所示。

```
from pyecharts import Overlap, Bar, Line, Grid
bar = Bar()
bar.add('房屋均价(元)', x, y1)
bar.add('人均工资(元)', x, y2)
line = Line()
line.add('房屋均价(元)', x, y1, is_label_show = True)
line.add('人均工资(元)', x, y2, is_label_show = True)
overlap = Overlap()
overlap.add(bar)
overlap.add(line)
overlap
```

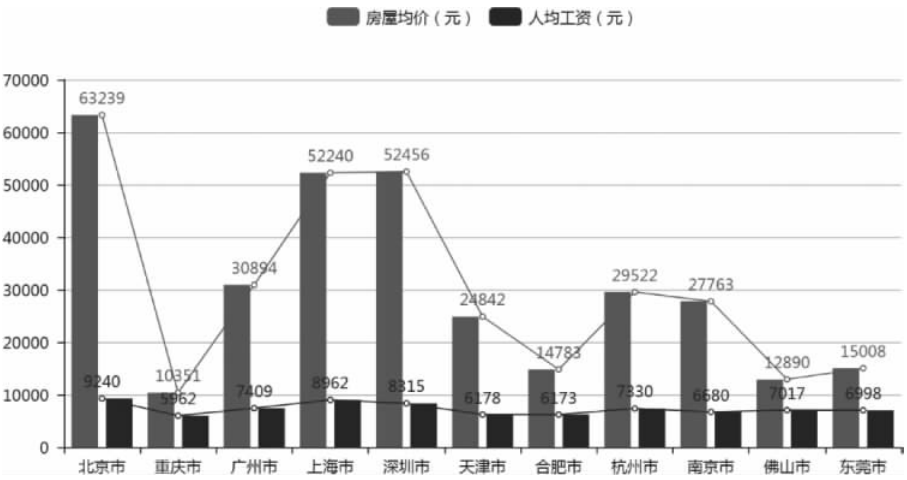


图 3-29　房价与工资.xlsx 的数据条形图叠加折线图

使用 pyecharts 也可创建仪表盘图，代码如下，结果如图 3-30 所示。

```
from pyecharts import Gauge
gauge = Gauge()
gauge.add('', '', '80', scale_range=[0, 100], angle_range=[180, 0])
gauge
```

图 3-30　pyecharts 创建的仪表盘图

使用 pyecharts 也可创建城市散点图，感兴趣的读者可以尝试运行，看看运行结果。

```
import numpy as np
temp = np.array(data[['城市','房屋均价']]).tolist()
a = []
for i in temp:
    z=tuple(i)
    a. append(z)
from pyecharts import Geo
```

```
geo = Geo("全国主要城市房价散点图", title_color = "#fff", title_pos = "center", width =
800, height = 600, background_color = '#404a59')
city, value = geo.cast(a)
geo.add("", city, value, visual_range = [10000, 50000], visual_text_color = "#fff",
symbol_size = 15, is_visualmap = True)
geo
```

使用 pyecharts 也可创建城市散点涟漪图，代码如下：

```
from pyecharts import Geo
geo = Geo("全国主要城市房价散点涟漪图", title_color = "#fff", title_pos = "center",
width = 800, height = 600, background_color = '#404a59')
city, value = geo.cast(a)
geo.add("", city, value, type = "effectScatter", is_random = True, effect_scale = 5)
geo
```

使用 pyecharts 也可创建热力图，代码如下：

```
from pyecharts import Geo
geo = Geo("全国主要城市房价热力图", title_color = "#fff", title_pos = "center", width =
800, height = 600, background_color = '#404a59')
city, value = geo.cast(a)
geo.add("", city, value, type = "heatmap", is_visualmap = True, visual_range = [10000,
20000], visual_text_color = '#fff')
geo
```

使用 pyecharts 也可创建全国省份地图，代码如下：

```
from pyecharts import Map
city = list(data['省份'])
value = list(data['房屋均价'])
map = Map( width = 800, height = 600)
map.add("", city, value, maptype = 'china', is_label_show = True, is_visualmap = True,
visual_range = [5000, 60000], visual_text_color = '#000')
map
```

使用 pyecharts 也可创建指定省份城市图，代码如下：

```
from pyecharts import Map
city = np.array(data['城市']).tolist()
value = np.array(data['房屋均价']).tolist()
map = Map( width = 800, height = 600)
map.add("",city,value,maptype = '广东', is_label_show = True, is_visualmap = True, visual_
range = [10000, 20000],visual_text_color = '#000')
```

```
map
```

从上面的各种图形中可以看出，使用 pyecharts 作的图确实很漂亮，以后做数据分析展示图的时候，可以尝试使用 pyecharts 进行作图，然后将这些图片放到 PPT 报告里，从而提高报告的展现力。

3.8 stats 简单统计分析

Python 虽然不像 R 语言那样，专为统计而生，但是它也可以进行统计分析与检验。scipy 库中的 stats 包就能实现大多数的统计功能。

学生的考试成绩会受到多种因素的影响，如：是否分场、试卷难易程度和临场发挥情况。本次仅考虑“是否分场”这一因素。下面以北京市日坛实验中学的初三生物考试成绩为例，分析在考场合并前后，学生的成绩有无显著差异。

```
import pandas as pd
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']   #指定默认字体
mpl.rcParams['axes.unicode_minus'] = False     #解决保存图像是负号'-'显示为方块的问题
import os
os.chdir('/Users/zhaikun/Desktop')
data = pd.read_excel('成绩.xls')
```

下面预览一下数据，如图 3-31 所示。

```
In [4]: data.head()
Out[4]:
```

	分场1	分场2	分场3	分场4	分场_月考	合场1	合场2	合场3	合场_期中
0	27	26	29	38	30	24	29	37	32
1	30	38	38	40	45	38	38	36	44
2	43	49	47	46	43	48	48	42	50
3	30	34	29	41	32	36	29	38	15
4	39	45	40	44	48	50	40	40	48

图 3-31　初三生物考试成绩数据预览图

首先计算出每个分场的考试成绩平均值，再计算合并考场之后的成绩平均值，代码如下所示。

```
data['分场平均值'] = data.ix[:,0:5].mean(1)
data['合场平均值'] = data.ix[:,5:9].mean(1)
temp = data[['分场平均值', '合场平均值']]
temp.plot()
plt.show()
```

计算考试成绩平均值之后，可以通过可视化，查看每个考生的成绩差异，结果如图 3-32 所示。

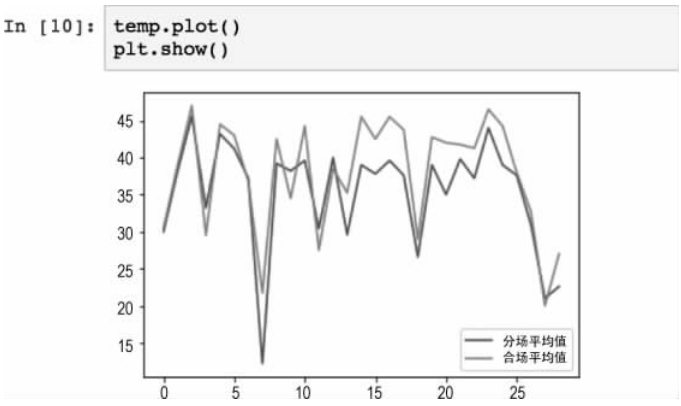


图 3-32　生物考试成绩差异图

由图 3-32 可以发现，合并考场之后，考试平均成绩明显比分场考试平均成绩高。

下面进行“相关样本的 t 检验”，原假设是样本均值相等，即考生在考场合并前后的考试成绩应该是相等的。代码如下所示，得到的结果如图 3-33 所示。

```
from scipy import stats
stats.ttest_rel(temp['分场平均值'], temp['合场平均值'])
```

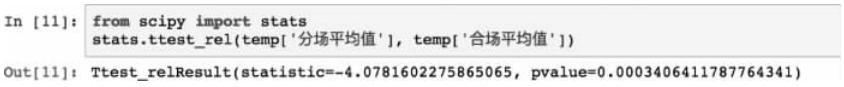


图 3-33　t 检验结果图

从计算结果中，可以发现 $P<0.05$，即原假设不成立，就是考场合并前后，考生的考试成绩均值差异显著。

stats 包除了可以进行 t 检验以外，还可以进行卡方检验、方差分析等统计上常用的检验手段，这里不进行详细介绍，感兴趣的读者可自行查阅相关资料。

3.9 小结

本章主要介绍了与 Python 语言相关的实际应用，这些应用均使用了第三方包，主要包括数据的连接、词云图、社交网络、JSON 解析、OCR 文字识别以及 pyecharts 绘图等。通过这些实际应用，展示了 Python 作为“胶水语言”的魅力，感兴趣的读者可以查找相关资料，进行进一步的学习与探索。