



中國人民大學
RENMIN UNIVERSITY OF CHINA

计算传播理论与实务

2019-2020秋季学期

第三讲 文本分析

Python实现详解

授课教师：范举副教授、塔娜讲师

时间：2019年11月11日

复习：Scherer 情感状态类型

- **情绪 (emotion)** : 有一定原因引发的同步反应
 - 例如悲伤 (sadness) , 快乐 (joy)
- **心情 (mood)** : 没有明显原因引发的长期低强度的主观感受变化
 - 例如忧郁 (gloomy) , 倦怠 (listless)
- **人际立场 (interpersonal stance)** : 对他人的特定反应
 - 例如疏远 (distant) , 冷漠 (cold)
- **态度 (attitude)** : 对特定人或事物的带有主观色彩的偏好或倾向
 - 喜欢 (like) , 讨厌 (hate)
- **个性特质 (personal traits)** : 相对稳定的个性倾向和行为趋势
 - 例如焦虑 (nervous) , 渴望 (anxious)

复习：什么是情感分析

- 情感分析是指从文本（或其它数据）中检测出人的态度的技术
 - 态度 (attitude) : 对特定人或事物的带有主观色彩的偏好或倾向
 - *enduring, affectively colored beliefs, dispositions towards objects or persons*

复习：什么是情感分析

- 情感分析有很多别名
 - 意见抽取 (Opinion extraction)
 - 意见挖掘 (Opinion mining)
 - 情感挖掘 (Sentiment mining)
 - 主观倾向分析 (Subjectivity analysis)

复习：商品评论情感分析



HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner
\$89 online, \$100 nearby ★★★★☆ 377 reviews

September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

Reviews

Summary - Based on 377 reviews

1 star 2 3 4 stars 5 stars

What people are saying

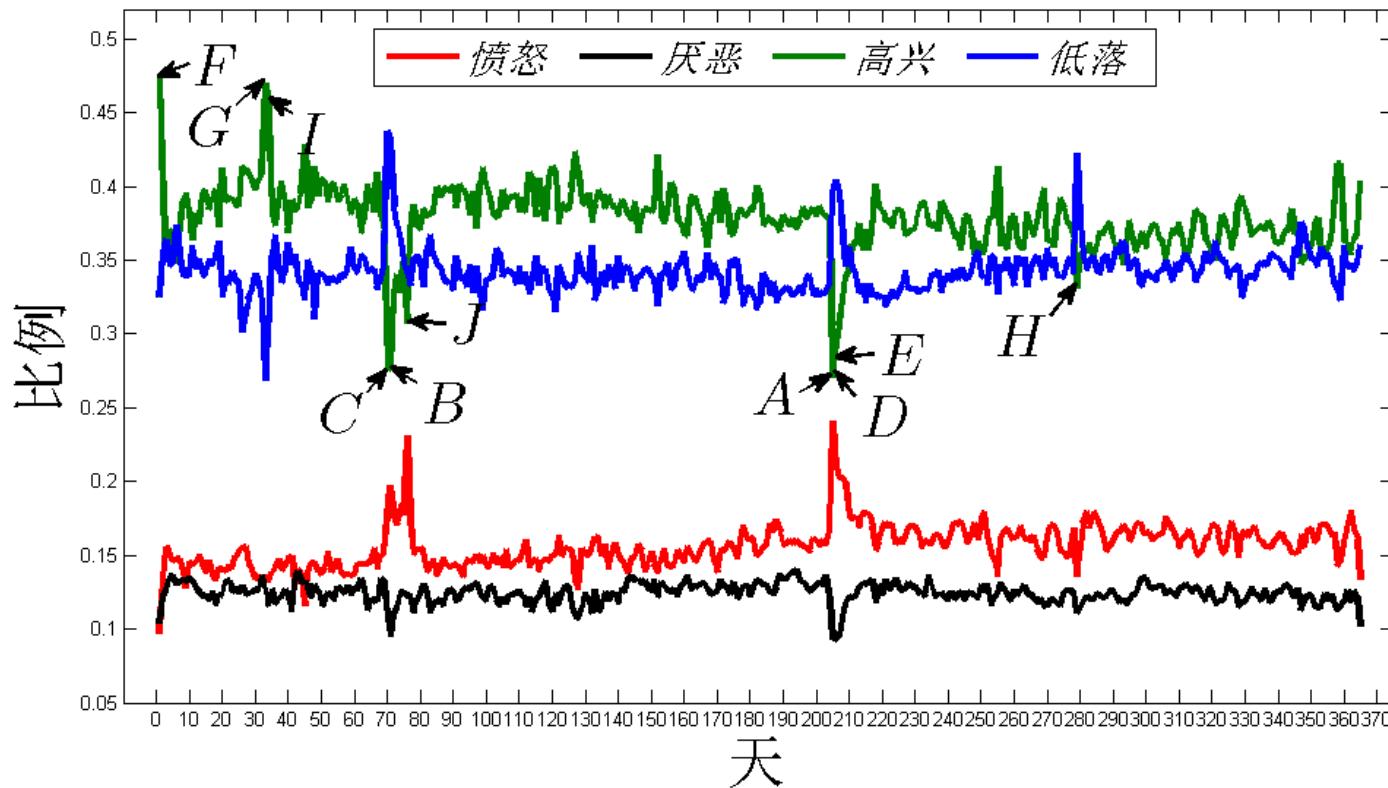
ease of use	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div>	"This was very easy to setup to four computers."
value	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div>	"Appreciate good quality at a fair price."
setup	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div>	"Overall pretty easy setup."
customer service	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div> <div style="width: 10px; height: 10px; background-color: green; border-radius: 5px;"></div>	"I DO like honest tech support people."
size	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div> <div style="width: 10px; height: 10px; background-color: green; border-radius: 5px;"></div>	"Pretty Paper weight."
mode	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div> <div style="width: 10px; height: 10px; background-color: green; border-radius: 5px;"></div>	"Photos were fair on the high quality mode."
colors	<div style="width: 10px; height: 10px; background-color: red; border-radius: 5px;"></div> <div style="width: 10px; height: 10px; background-color: green; border-radius: 5px;"></div>	"Full color prints came out with great quality."



复习：饭店评论情感分析

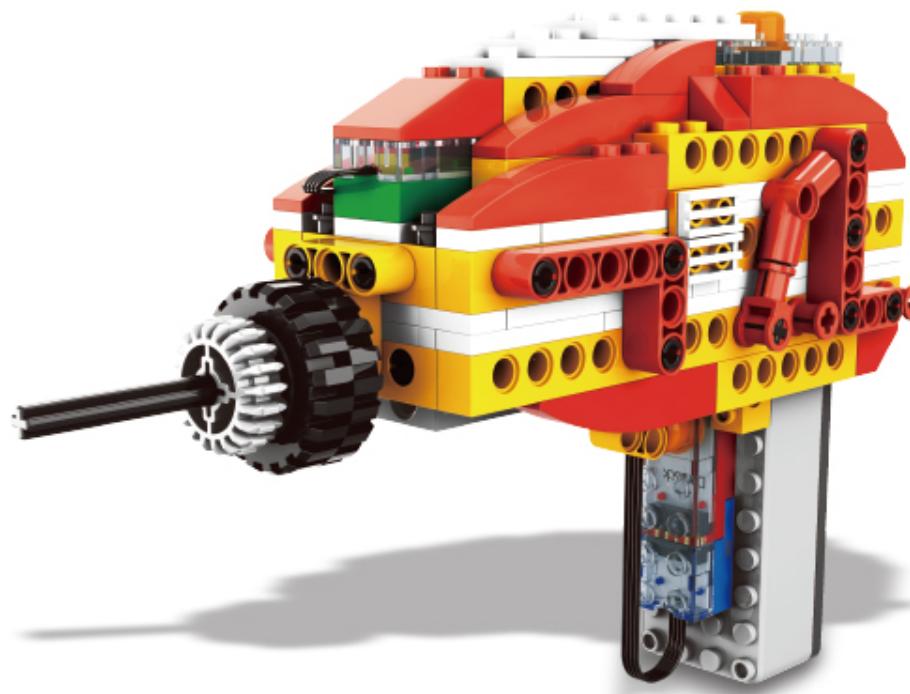
评论	打分
口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...	2
菜品丰富质量好，服务也不错！很喜欢！	4
说真的，不晓得有人排队的理由，香精香精香精，拜拜！	2
菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...	5
先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...	4
哎，冲着推荐来的，后悔死了，上菜巨慢不说，服务态度还超级差，东西咸的要死，而且普遍的贵，谁来...	1
超级差评????上菜慢??服务态度超级差、先把汤上来喝饱了??完事菜就上了一个，找服务员催半...	1

复习：微博情感分析



- 北航的先进网络分析研究小组(GANA) 对收集到的2011年的近7000万条微博进行情感分析

模块化编程



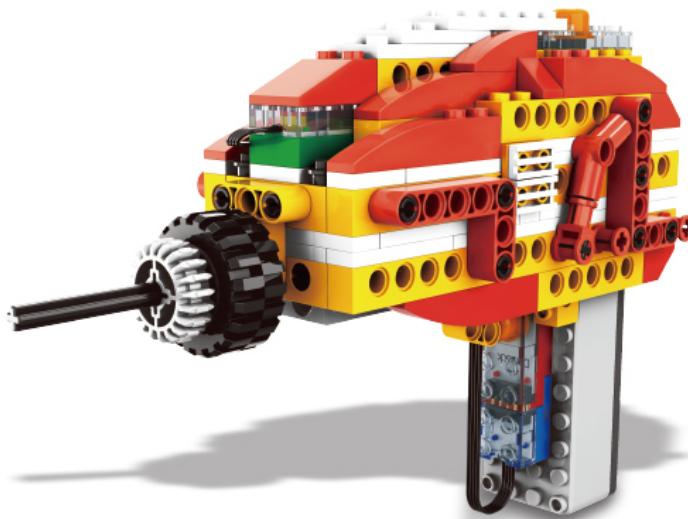


瓦萨号沉没原因

- 客户频繁更改需求
 - 在“瓦萨”号的骨架已经安装好的时候，国王下令增加战舰的长度
 - 得知了丹麦建成双层炮舰，国王把它改建成“双层”炮舰
- 程序测试像个闹剧
 - 让30个船员从船一端跑到另一端，以此检测船的摇动情况。
 - 试验中“瓦萨”号发生了危险的摇动，但对这个预警信号，却视而不见

模块化编程

- 客户的需求是多种多样的
 - 将一些基本功能封装成模块，开箱即用
- 程序测试非常重要
 - 分模块独立进行测试，避免问题的积攒



- 第三方类库(**Library**)
 - 专门研发出来供开发人员调用，以实现特定功能的工具集

常用的Python第三方类库

- Pandas: 数据操作与分析
- jieba: 中文分词
- scikit-learn: 机器学习算法
- Matplotlib: 图表绘制

本讲
重点

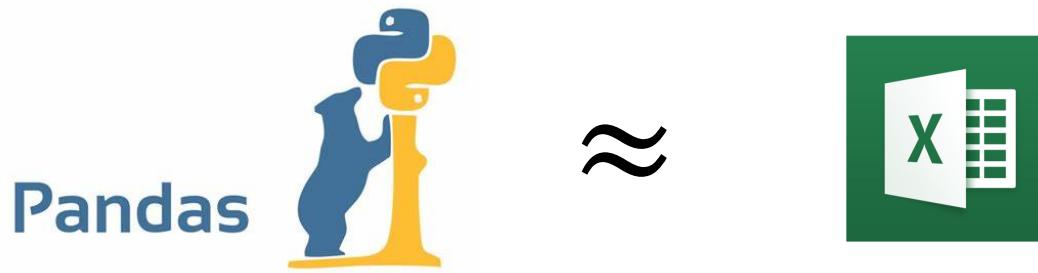
- NLTK: 自然语言处理工具包
- Keras: 深度学习 (神经网络)
- NetworkX: 网络数据操作与算法

第3.5.1节

Pandas介绍与实例

Pandas简介

- Pandas是目前最流行的Python数据分析类库
 - Pandas → Python Data Analysis Library
- 如何在直观上理解Pandas的功能?



- pandas适合于许多不同类型的数据，包括
 - 表格数据：表格的各个列，可以具有不同的类型
 - 时间序列数据：有序、和无序的时间序列数据的处理
 - 矩阵：异构数据类型矩阵，可以设定行和列的标签

开始使用Pandas

- 安装
 - 通过pip来执行安装

```
sudo pip3 install pandas
```

- 通过conda执行安装

```
conda install pandas
```

- 开始使用

```
import pandas as pd  
import numpy as np
```

理解Pandas的核心数据结构

- Series

- 一维数组，与Numpy中的一维数组array类似，二者与Python基本的数据结构List也很相近。
- 区别：List中的元素可以是不同的数据类型，而Array和Series中则只允许存储相同数据类型的元素

Team	Win	Loss	Win%
Houston Rockets	20	4	0.83
Golden State Warriors	21	6	0.78
San Antonio Spurs	19	8	0.7
Minnesota Timberwolves	16	11	0.59
Denver Nuggets	14	12	0.54
Portland Trail Blazers	13	12	0.52
New Orleans Pelicans	14	13	0.52
Utah Jazz	13	14	0.48

```
win_col = pd.Series ([20,21,19,16,14,13,  
                     14,13])  
  
print (win_col)  
  
0    20  
1    21  
2    19  
3    16  
4    14  
5    13  
6    14  
7    13  
dtype: int64
```

练习：为其它列创建相应的Series

理解Pandas的核心数据结构

- Series

- 一维数组，与Numpy中的一维数组array类似，二者与Python基本的数据结构List也很相近。
- 区别：List中的元素可以是不同的数据类型，而Array和Series中则只允许存储相同数据类型的元素

```
win_col = pd.Series ([20,21,19,np.nan,14,13,  
                     14,13])  
print (win_col)
```

```
0    20.0  
1    21.0  
2    19.0  
3    NaN  
4    14.0  
5    13.0  
6    14.0  
7    13.0  
dtype: float64
```

- 处理缺失值

- NaN表示Not a Number
- np.nan

理解Pandas的核心数据结构

- DataFrame

- 二维的表格型数据结构
- 可以将DataFrame理解为Series的容器
- 每一列的数据类型都是相同的，而不同的列可以是不同的数据类型

Team	Win	Loss	Win%
Houston Rockets	20	4	0.83
Golden State Warriors	21	6	0.78
San Antonio Spurs	19	8	0.7
Minnesota Timberwolves	16	11	0.59
Denver Nuggets	14	12	0.54
Portland Trail Blazers	13	12	0.52
New Orleans Pelicans	14	13	0.52
Utah Jazz	13	14	0.48

```
nba_df = pd.DataFrame ({'Team': team_col,
                        'Win': win_col,
                        'Loss': loss_col})
print (nba_df)
```

	Team	Win	Loss
0	Houston Rockets	20	4
1	Golden State Warriors	21	6
2	San Antonio Spurs	19	8
3	Minnesota Timberwolves	16	11
4	Denver Nuggets	14	12
5	Portland Trail Blazers	13	12
6	New Orleans Pelicans	14	13
7	Utah Jazz	13	14

行标签

理解Pandas的核心数据结构

• DataFrame

```
nba_df.head() # 前五行
```

	Team	Win	Loss
0	Houston Rockets	20	4
1	Golden State Warriors	21	6
2	San Antonio Spurs	19	8
3	Minnesota Timberwolves	16	11
4	Denver Nuggets	14	12

```
nba_df.tail() # 后五行
```

	Team	Win	Loss
3	Minnesota Timberwolves	16	11
4	Denver Nuggets	14	12
5	Portland Trail Blazers	13	12
6	New Orleans Pelicans	14	13
7	Utah Jazz	13	14

```
nba_df.describe() # 描述信息  
print (nba_df.index) # 行标签  
print (nba_df.columns) # 列标签  
print (nba_df.dtypes) # 各列的数据类型  
print (nba_df.values) # 具体的值
```

```
RangeIndex(start=0, stop=8, step=1)  
Index(['Team', 'Win', 'Loss'], dtype='object')  
Team      object  
Win       int64  
Loss      int64  
dtype: object  
[['Houston Rockets' 20 4]  
 ['Golden State Warriors' 21 6]  
 ['San Antonio Spurs' 19 8]  
 ['Minnesota Timberwolves' 16 11]  
 ['Denver Nuggets' 14 12]  
 ['Portland Trail Blazers' 13 12]  
 ['New Orleans Pelicans' 14 13]  
 ['Utah Jazz' 13 14]]
```

理解Pandas的核心数据结构

- 其它数据结构（了解即可）
 - Time Series：以时间为索引的Series
 - Panel：三维数组，可以理解为DataFrame的容器
- 这些数据结构可以表示和处理金融、统计、社会科学、以及众多的工程领域的大多数应用场景用到的数据。

Pandas的常用功能函数

- 标签排序
 - 针对行标签或列标签对数据进行排序

```
df1 = nba_df.sort_index (axis=0, ascending=False)  
df1.head()
```

	Team	Win	Loss
7	Utah Jazz	13	14
6	New Orleans Pelicans	14	13
5	Portland Trail Blazers	13	12
4	Denver Nuggets	14	12
3	Minnesota Timberwolves	16	11

```
df2 = nba_df.sort_index (axis=1, ascending=False)  
df2.head()
```

	Win	Team	Loss
0	20	Houston Rockets	4
1	21	Golden State Warriors	6
2	19	San Antonio Spurs	8
3	16	Minnesota Timberwolves	11
4	14	Denver Nuggets	12

Pandas的常用功能函数

- 数据排序
 - 基于某个列，对DataFrame进行排序

```
nba_df = nba_df.sort_values(by='Win')
nba_df.head()
```

	Team	Win	Loss
5	Portland Trail Blazers	13	12
7	Utah Jazz	13	14
4	Denver Nuggets	14	12
6	New Orleans Pelicans	14	13
3	Minnesota Timberwolves	16	11

Pandas的常用功能函数

- 提取部分数据
 - 通过列名，提取一个数据列

```
print (nba_df[ 'Team' ])
```

```
5      Portland Trail Blazers
7              Utah Jazz
4          Denver Nuggets
6      New Orleans Pelicans
3      Minnesota Timberwolves
2          San Antonio Spurs
0          Houston Rockets
1      Golden State Warriors
Name: Team, dtype: object
```

Pandas的常用功能函数

- 提取部分数据
 - 通过列名，提取一个数据列
 - 可以通过行号范围，提取部分数据
 - 提取一个数据块

```
print (nba_df.iloc[1:5,0:2])
print (nba_df.iloc[1,1])
```

	Team	Win
7	Utah Jazz	13
4	Denver Nuggets	14
6	New Orleans Pelicans	14
3	Minnesota Timberwolves	16
13		

Pandas的常用功能函数

- 提取部分数据
 - 通过列名，提取一个数据列
 - 可以通过行号范围，提取部分数据
 - 提取一个数据块
 - 使用条件过滤，获取部分行数据

```
print ("提取Win大于16的数据")
print (nba_df[nba_df.Win>16])
print ("提取某些球队的数据")
print (nba_df[nba_df.Team.isin(['Utah Jazz'])])
```

提取Win大于16的数据

	Team	Win	Loss
2	San Antonio Spurs	19	8
0	Houston Rockets	20	4
1	Golden State Warriors	21	6

提取某些球队的数据

	Team	Win	Loss
7	Utah Jazz	13	14

Pandas的常用功能函数

- 计算每列的均值
 - 通过DataFrame的mean方法，可以计算每个数据列的均值

```
print (nba_df.mean())
```

```
Win      16.25
Loss     10.00
dtype: float64
```

Pandas的常用功能函数

- 利用已有列生成新的列

```
nba_df[ "Win%" ]=nba_df[ 'Win' ]/(nba_df[ 'Win' ]+nba_df[ 'Loss' ])
nba_df.head()
nba_df.sort_values(by='Win%', ascending=False)
```

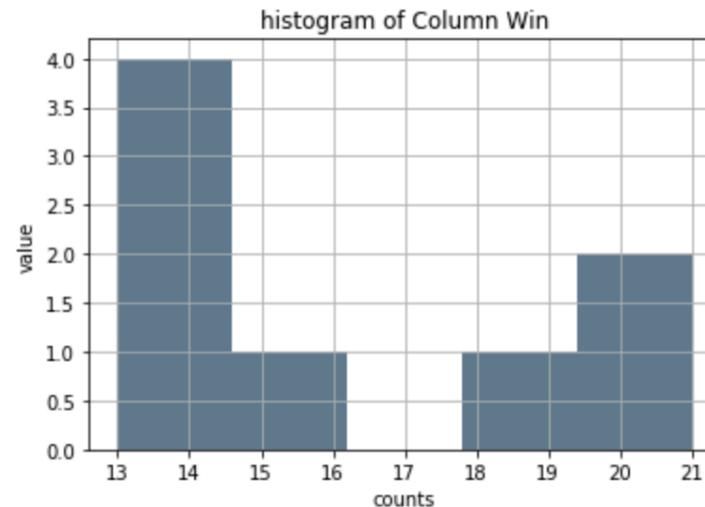
	Team	Win	Loss	Win%
0	Houston Rockets	20	4	0.833333
1	Golden State Warriors	21	6	0.777778
2	San Antonio Spurs	19	8	0.703704
3	Minnesota Timberwolves	16	11	0.592593
4	Denver Nuggets	14	12	0.538462
5	Portland Trail Blazers	13	12	0.520000
6	New Orleans Pelicans	14	13	0.518519
7	Utah Jazz	13	14	0.481481

Pandas的常用功能函数

- 计算直方图
 - 直方图是数据集里各个取值的频率的图形表示

```
print (nba_df.Win.value_counts())
import matplotlib.pyplot as plt
nba_df.Win.plot.hist(grid=True, bins=5, color="#607c8e")
plt.title('histogram of Column Win')
plt.xlabel('counts')
plt.ylabel('value')
plt.show()
```

```
14      2
13      2
20      1
19      1
21      1
16      1
Name: Win, dtype: int64
```



Pandas的常用功能函数

- 字符串处理
 - 可以对字符串类型的列进行批量处理
 - 例如：将字符串编程小写形式

```
nba_df['Team'] = nba_df['Team'].str.lower()  
nba_df.head()
```

	Team	Win	Loss	Win%
5	portland trail blazers	13	12	0.520000
7	utah jazz	13	14	0.481481
4	denver nuggets	14	12	0.538462
6	new orleans pelicans	14	13	0.518519
3	minnesota timberwolves	16	11	0.592593

Pandas的常用功能函数

- 对每列运用一个函数
 - 可以对 DataFrame 的每个数据列，运用某个函数

```
nba_df['Win'].apply(lambda x: x-1)
```

```
5    12  
7    12  
4    13  
6    13  
3    15  
2    18  
0    19  
1    20  
Name: Win, dtype: int64
```

```
def make_label (win_p):  
    if win_p > 0.5:  
        return 'good'  
    else:  
        return 'bad'  
nba_df['label'] = nba_df['Win%'].apply(make_label)  
nba_df.head()
```

	Team	Win	Loss	Win%	label
5	portland trail blazers	13	12	0.520000	good
7	utah jazz	13	14	0.481481	bad
4	denver nuggets	14	12	0.538462	good
6	new orleans pelicans	14	13	0.518519	good
3	minnesota timberwolves	16	11	0.592593	good

Pandas的常用功能函数

- 文件读/写
 - 利用pandas提供的函数，我们可以把数据保存到文件中，也可以从文件中读取数据。Pandas支持的数据文件格式，包括CSV、HDF5、Excel等。

```
nba_df.to_csv('datasets/nba_out.csv')
nba_df2 = pd.read_csv('datasets/nba.csv')
nba_df2.head()
```

	Team	Win	Loss	Win%
0	Houston Rockets	20	4	0.83
1	Golden State Warriors	21	6	0.78
2	San Antonio Spurs	19	8	0.70
3	Minnesota Timberwolves	16	11	0.59
4	Denver Nuggets	14	12	0.54

Pandas的其它功能

- 处理数据的缺失值(Missing Data)，包括浮点数和非浮点数的缺失值。
- 动态扩展性，用户可以插入或者删除DataFrame数据结构的列。
- 数据对齐(Data Alignment)：用户可以把数据对象对齐到标签(Label)，或者由Series、DataFrame等数据结构自行对齐。
- 组聚集功能(Group by and Aggregation)。
- 把其它NumPy等第三方库的数据结构转换成DataFrame的功能。
- 数据转换(Transformation)。

Pandas的其它功能

- 数据集的合并(Merging)和连接Joining)。
- 从CSV文件、Excel文件、数据库进行装载数据，把数据保存到HDF5格式的文件，以及从HDF5格式的文件装载数据。
- 坐标轴的层次标签(Hierarchical Labeling)。
- 数据透视表的旋转(Pivoting)、改变形状(Reshaping)。
- 对大数据集进行基于标签的(Label based)数据切片(Slicing)、提取子集(Sub Setting)、建立和使用索引(Fancy Indexing)。
- Pandas还提供面向时间序列数据处理的一些特殊功能，比如时间频率转换、移动窗口上的统计值计算、移动窗口上的线性回归等。

程序参考

sklearn-sa.ipynb

- 使用pandas进行数据准备
 - 从csv文件中读入大众点评数据
 - 根据分数准备sentiment属性
 - 使用apply函数进行简单情感分析
 - 使用apply函数进行分词

读取数据

- 如何从CSV文件中读取数据?
 - 使用pandas !

```
In [2]: data = pd.read_csv('datasets/dianping.csv')
data.head()
```

Out[2]:

		comment	star
0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...		2
1	菜品丰富质量好，服务也不错！很喜欢！		4
2	说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！		2
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...		5
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...		4

数据预处理

- 根据star属性设置sentiment属性
 - 三星以上 → positive；三星及以下 → negative

```
In [3]: def make_label(star):
    if star > 3:
        return 1
    else:
        return 0

data['sentiment'] = data.star.apply(make_label)
data = data[['comment', 'sentiment']]
data.head()
```

Out[3]:

	comment	sentiment
--	---------	-----------

0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...	0
1	菜品丰富质量好，服务也不错！很喜欢！	1
2	说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！	0
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...	1
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...	1

小白版情感分析：snownlp库

```
In [4]: from snownlp import SnowNLP

text1 = '这个东西不错'
text2 = '这个东西很垃圾'

s1 = SnowNLP(text1)
s2 = SnowNLP(text2)

print(s1.sentiments,s2.sentiments)

def snow_result(comemnt):
    s = SnowNLP(comemnt)
    if s.sentiments >= 0.6:
        return 1
    else:
        return 0

data['snlp_result'] = data.comment.apply(snow_result)
data.head()
```

```
0.8623218777387431 0.21406279508712744
```

Out[4]:

	comment	sentiment	snlp_result
0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...	0	0
1	菜品丰富质量好，服务也不错！很喜欢！	1	1
2	说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！	0	1
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...	1	0
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...	1	0

小白版情感分析：snownlp库

Out[4]:

		comment	sentiment	snlp_result
0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...		0	0
1		菜品丰富质量好，服务也不错！很喜欢！	1	1
2		说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！	0	1
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...		1	0
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...		1	0

- 使用snownlp库的优缺点
 - 优点：简单方便
 - 缺点：准确率低

```
from snownlp import SnowNLP
text1 = '这个东西不错'
text2 = '这个东西很垃圾'
s1 = SnowNLP(text1)
s2 = SnowNLP(text2)
print(s1.sentiments,s2.sentiments)
```

第3.5.2节

Jieba介绍与实例

Jieba分词

```
#encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学",cut_all=True)
print ("Full Mode:", "/ ".join(seg_list)) #全模式

seg_list = jieba.cut("我来到北京清华大学",cut_all=False)
print ("Default Mode:", "/ ".join(seg_list)) #精确模式

seg_list = jieba.cut("他来到了网易杭研大厦") #默认是精确模式
print (" ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所, 后在日本京都大学深造") #搜索引擎模式
print (" ".join(seg_list))
```

```
Building prefix dict from the default dictionary ...
Dumping model to file cache /var/folders/8d/621h93xj51v9v1r3f8qg6m6r0000gn/T/jieba.cache
Loading model cost 1.082 seconds.
Prefix dict has been built successfully.
```

Full Mode: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

Default Mode: 我/ 来到/ 北京/ 清华大学

他, 来到, 了, 网易, 杭研, 大厦

小明, 硕士, 毕业, 于, 中国, 科学, 学院, 科学院, 中国科学院, 计算, 计算所, , , 后, 在, 日本, 京都, 大学, 日本京都大学, 深造

- 精确模式，试图将句子最精确地切开，适合文本分析；
- 全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；
- 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

Jieba分词

- 基本分词函数
 - 调用jieba.cut (str, cut_all=False) # 默认精确模式
 - 返回的结构都是一个可迭代的generator
- 字符串join()函数
 - 语法： 'sep'.join(seq)
 - 参数说明
 - sep： 分隔符。可以为空
 - seq： 要连接的元素序列、字符串、元组、字典
 - 上面的语法即： 以sep作为分隔符， 将seq所有的元素合并成一个新的字符串
 - 返回值： 返回一个以分隔符sep连接各个元素后生成的字符串

分词：准备词袋模型

- 使用jieba分词进行简单分词

```
def simple_word_cut (texts):
    return ' '.join(jieba.cut(texts, cut_all=False))

data['simple_cut_comment'] = data.comment.apply(simple_word_cut)
data.head()
```

```
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/8d/62lh93xj51v9v1r3f8qg6m6r0000gn/T/jieba.cache
Loading model cost 0.975 seconds.
Prefix dict has been built successfully.
```

	comment	sentiment	simple_cut_comment
0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...	0	口味： 不知道 是我口高了， 还是这家真不怎么样。 ?? 我感觉 口...
1	菜品丰富质量好，服务也不错！很喜欢！	1	菜品 丰富 质量 好， 服务 也 不错！ 很 喜欢！
2	说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！	0	说真的， 不晓得 有人 排队 的 理由， 香精 香精 香精 香精， 拜拜！
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...	1	菜量 实惠， 上菜 还 算 比较 快， 疙瘩汤 喝 出 了 秋 日 的 暖 意， 烧 茄子 吃...
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...	1	先说 我 算是 娜娜 家风 荷园 开业 就 一直 在 这里 吃 ?? 每次 出去 回 来 总...

去除停用词 (stopwords)

In [9]:

```
def word_cut (texts):
    words_list = []
    word_generator = jieba.cut(texts, cut_all=False) # 返回的是一个迭代器
    with open('hit_stopwords.txt') as f:
        str_text = f.read()
    for word in word_generator:
        if word.strip() not in str_text:
            words_list.append(word)
    #print ('1')
    return ' '.join(words_list) # 注意是空格

data['cut_comment'] = data.comment.apply(word_cut)

data.head()
```

Out[9]:

	comment	sentiment	cut_comment
0	口味：不知道是我口高了，还是这家真不怎么样。??我感觉口味确实很一般很一般。上菜相当快，我敢...	0	口味 知道 我口 高 这家 不怎么样 感觉 口味 确实 很 很 上菜 相当 快 我敢 菜 都...
1	菜品丰富质量好，服务也不错！很喜欢！	1	菜品 丰富 质量 服务 不错 很 喜欢
2	说真的，不晓得有人排队的理由，香精香精香精香精，拜拜！	0	说真的 晓得 有人 排队 理由 香精 香精 香精 香精 拜拜
3	菜量实惠，上菜还算比较快，疙瘩汤喝出了秋日的暖意，烧茄子吃出了大阪烧的味道，想吃土豆片也是口...	1	菜量 实惠 上菜 算 比较 快 疙瘩汤 喝出 秋日 暖意 烧茄子 吃 出 大阪 烧 味道 想...
4	先说我算是娜娜家风荷园开业就一直在这里吃??每次出去回来总想吃一回??有时觉得外面的西式简餐...	1	先说 算是 娜娜 家风 荷园 开业 吃 每次 出去 回来 总想 吃 一回 有时 觉得 外面 ...

第3.5.3节

SCIKIT-LEARN 介绍与实例

Scikit-learn类库简介



- Scikit-learn是面向Python的机器学习软件包
- Scikit-learn软件包支持主流的有监督学习方法与无监督学习方法
 - 有监督的学习方法，包括通用的线性模型(Generalized Linear Model, 比如线性回归Linear Regression)、支持向量机(Support Vector Machine, SVM)、决策树(Decision Tree)、贝叶斯方法(Bayesian Method等)
 - 无监督学习方法，包括聚类(Clustering)、因子分析(Factor Analysis)、主成份分析(Principal Component Analysis)、无监督神经网络(Unsupervised Neural Network)等。

使用sklearn进行文本分析

- 步骤I：数据准备

```
from sklearn.model_selection import train_test_split

corpus = ['口味 喜欢', '不错 喜欢', '喜欢 喜欢', '菜量 实惠', '不错',
          '不怎么样', '后悔', '不喜欢', '难吃', '差评']
label = [1,1,1,1,1, 0,0,0,0,0]
X_train, X_test, y_train, y_test = train_test_split(corpus, label,
                                                    test_size=0.4)

print ('X_train: ', X_train)
print ('y_train: ', y_train)
print ('X_test: ', X_test)
print ('y_test: ', y_test)
```

```
X_train:  ['喜欢 喜欢', '后悔', '不怎么样', '差评', '难吃', '菜量 实惠']
y_train:  [1, 0, 0, 0, 0, 1]
X_test:   ['不错 喜欢', '不喜欢', '不错', '口味 喜欢']
y_test:   [1, 0, 1, 1]
```

使用sklearn进行文本分析

- 步骤2：提取特征

```
from sklearn.feature_extraction.text import CountVectorizer
cvect = CountVectorizer()
X_train_c = cvect.fit_transform(X_train)
print(cvect.get_feature_names())
print(X_train_c.toarray())
```

```
['不怎么样', '后悔', '喜欢', '实惠', '差评', '菜量', '难吃']
[[0 0 2 0 0 0]
 [0 1 0 0 0 0]
 [1 0 0 0 0 0]
 [0 0 0 0 1 0]
 [0 0 0 0 0 1]
 [0 0 0 1 0 1]]
```

CountVectorizer的详细定义

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

使用sklearn进行文本分析

- 步骤2：提取特征

```
from sklearn.feature_extraction.text import TfidfVectorizer
tvect = TfidfVectorizer()
X_train_t = tvect.fit_transform(X_train)
print(tvect.get_feature_names())
print(X_train_t.toarray())
```

```
['不怎么样', '后悔', '喜欢', '实惠', '差评', '菜量', '难吃']
[[0.          0.          1.          0.          0.          0.
  0.          ]
 [0.          1.          0.          0.          0.          0.
  0.          ]
 [1.          0.          0.          0.          0.          0.
  0.          ]
 [0.          0.          0.          0.          1.          0.
  0.          ]
 [0.          0.          0.          0.          0.          0.
  1.          ]
 [0.          0.          0.          0.          0.          0.
  0.          ]
 [0.          0.          0.          0.70710678 0.          0.70710678
  0.          ]]
```

TfidfVectorizer的详细定义

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

使用sklearn进行文本分析

- 步骤3：分类

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train_t, y_train)

train_accuracy = nb.score(X_train_t, y_train)
print (train_accuracy)

X_test_t = tvect.transform(X_test)
print(tvect.get_feature_names())
print (X_test_t.toarray())
y_predict = nb.predict(X_test_t)
print ('y_predict: ', y_predict)
print ('y_test', y_test)
```

```
1.0
['不怎么样', '后悔', '喜欢', '实惠', '差评', '菜量', '难吃']
[[0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0.]]
y_predict: [1 0 0 1]
y_test [1, 0, 1, 1]
```

使用sklearn进行文本分析

- 步骤4：评测指标

```
from sklearn import metrics
print(metrics.classification_report(y_test, y_predict))
metrics.confusion_matrix(y_test, y_predict)
```

	precision	recall	f1-score	support
0	0.50	1.00	0.67	1
1	1.00	0.67	0.80	3
micro avg	0.75	0.75	0.75	4
macro avg	0.75	0.83	0.73	4
weighted avg	0.88	0.75	0.77	4

TN FP
array([[1, 0],
 [1, 2]])
FN TP

计算Precision和Recall

以标签1为positive的时候

程序参考

sklearn-sa.ipynb

- 使用sklearn进行情感分析

提取特征

```
: X = data['cut_comment']
y = data['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

: from sklearn.feature_extraction.text import TfidfVectorizer

vect = TfidfVectorizer(max_df = 0.8,
                      min_df = 3,
                      token_pattern=u'(?u)\\b[^\\d\\w]\\w+\\b'
                     )

features = pd.DataFrame(vect.fit_transform(X_train).toarray(), columns=vect.get_feature_names()
                        .head()
```

	app	ipad	ok	ps	wifi	一下	一个个	一个半	一个多	一人	...	麻将	麻烦	麻辣	麻酱	黄瓜	黄盖	黄色	黑椒	默默	齐全
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1910 columns

使用朴素贝叶斯分类器

```
In [13]: from sklearn.naive_bayes import MultinomialNB  
  
nb = MultinomialNB()  
  
X_train_vect = vect.fit_transform(X_train)  
nb.fit(X_train_vect, y_train)  
  
train_accuracy = nb.score(X_train_vect, y_train)  
print (train_accuracy)  
  
0.914375
```

```
In [14]: X_test_vect = vect.transform(X_test)  
test_accuracy = nb.score(X_test_vect, y_test)  
  
y_predict = nb.predict(X_test_vect)  
  
print('测试准确率', test_accuracy)  
  
from sklearn.metrics import classification_report  
print("测试集上其他指标: \n",classification_report(y_test, y_predict))
```

测试准确率 0.82

测试集上其他指标:

	precision	recall	f1-score	support
0	0.85	0.77	0.81	197
1	0.79	0.87	0.83	203
micro avg	0.82	0.82	0.82	400
macro avg	0.82	0.82	0.82	400
weighted avg	0.82	0.82	0.82	400

复习：朴素贝叶斯

	docID	words in document	in $c = China?$
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

$$c_{\text{map}} = \arg \max_{c \in \mathcal{C}} \hat{P}(c|d) = \arg \max_{c \in \mathcal{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

训练过程 - I

- 步骤1：提取词典 (Vocabulary)
 - $V = \{\text{Chinese}, \text{Beijing}, \text{Shanghai}, \text{Macao}, \text{Tokyo}, \text{Japan}\}$
- 步骤2：计算训练文档个数
 - $N = 4$
- 步骤3：针对正例文档，表示为 c
 - 计算正例文档个数 $N_c = 3$
 - 计算先验概率 $P(c) = \frac{3}{4} = 0.75$
 - 将正例文档进行合并形成文本 text_c

训练过程 - 2

- 步骤3：针对正例文档，即 $c=China$
 - 针对词典 V 中的每个单词，进行如下计算

$$\bullet P(Chinese|c) = \frac{5+1}{8+6} = \frac{3}{7}$$

$$\bullet P(Beijing|c) = \frac{1+1}{8+6} = \frac{1}{7}$$

$$\bullet P(Shanghai|c) = \frac{1+1}{8+6} = \frac{1}{7}$$

$$\bullet P(Macao|c) = \frac{1+1}{8+6} = \frac{1}{7}$$

$$\bullet P(Tokyo|c) = \frac{0+1}{8+6} = \frac{1}{14}$$

$$\bullet P(Japan|c) = \frac{0+1}{8+6} = \frac{1}{14}$$

训练过程 - 3

- 步骤4：针对负例文档，表示为 \bar{c}

- 计算先验概率 $P(\bar{c}) = \frac{1}{4} = 0.25$

- 将负例文档进行合并形成文本text $_{\bar{c}}$

- $P(Chinese|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$

- $P(Beijing|\bar{c}) = \frac{0+1}{3+6} = \frac{1}{9}$

- $P(Shanghai|\bar{c}) = \frac{0+1}{3+6} = \frac{1}{9}$

- $P(Macao|\bar{c}) = \frac{0+1}{3+6} = \frac{1}{9}$

- $P(Tokyo|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$

- $P(Japan|\bar{c}) = \frac{1+1}{3+6} = \frac{2}{9}$

测试（预测）过程

- 给定测试文本
 - Chinese Chinese Chinese Tokyo Japan
- 计算正例的后验概率
 - $P(c|d) = \frac{3}{4} \cdot \left(\frac{3}{7}\right)^3 \cdot \frac{1}{14} \cdot \frac{1}{14} \approx 0.0003$
- 计算负例的后验概率
 - $P(\bar{c}|d) = \frac{1}{4} \cdot \left(\frac{2}{9}\right)^3 \cdot \frac{2}{9} \cdot \frac{2}{9} \approx 0.0001$
- 由于 $P(c|d) > P(\bar{c}|d)$, 预测测试文本的类别为正例
- 请你计算以下文本
 - Beijing Tokyo Japan

第3.5.4节

WORDCLOUD 介绍与实例

数据查看：绘制词云

- 绘制所有positive文档包含单词的词云

```
In [6]: def draw_wordcloud (words, color = 'white'):  
    wordcloud = WordCloud(stopwords = STOPWORDS,  
                          font_path="HYQiHei-105.ttf",  
                          background_color=color,  
                          width=2500,  
                          height=2000  
                      ).generate(words)  
    plt.figure(1,figsize=(13, 13))  
    plt.imshow(wordcloud)  
    plt.axis('off')  
    plt.show()  
  
    print("Positive words")  
  
data_pos = data[ data['sentiment'] == 1 ]  
words_pos = ' '.join(data_pos['cut_comment'])  
draw_wordcloud (words_pos)
```

数据查看：绘制词云

- 绘制所有positive文档包含单词的词云



数据查看：绘制词云

- 绘制所有negative文档包含单词的词云

