第一章 自动化数据采集及清洗 2：数据处理

主讲：方杰、李烜

福建江夏学院金融学院

本章目录

数据处理

数据处理是对数据按照所需的目标或要求进行特殊处理，最终得到标准、干净、完整、符合使用要求的数据，以便后续进一步使用。

数据清洗的主要工作包括缺失数据处理、数据格式规范、异常数据处理、重复数据处理等。

基于 Python 内置的 Pandas 包进行数据的处理。

数据缺失值处理

数据缺失是真实数据集中普遍存在的问题。

对于缺失值的处理，最简单的方法是忽略或直接删除有缺失的数据，但这种方式会造成数据的丢失或不完整，可能使后续的统计分析产生偏差。

缺失值判断

当数据量较小时，通过print函数打印数据并观察结果，其中显示为“NaN”(Not a Number) 的，即为缺失值，也称空值（NaN)。

对 DataFrame 对象可以使用isnull()或notnull()函数检测数据中是否含有空值

1. isnull() 判断 DataFrame 对象是否存在空值，并返回结果（True 或 False)
2. notnull() 判断 DataFrame 对象是否不存在空值，并返回结果 (True 或 False)
3. isnull().any() 判断一列数据的任意一个是否存在空值
4. isnull().all() 判断一列数据是否全部是空值

缺失值判断 (cont.)

对一列数据进行判断

- `isnull().any()` 或者 `isnull().any(axis=0)`
- `isnull().all()` 或者 `isnull().all(axis=0)`

对一行数据进行判断

- `isnull().any(axis=1)`
- `isnull().all(axis=1)`

缺失值处理——删除

Pandas 中dropna()函数可用来删除缺失值所在的行或列

```
df.dropna(axis=0, how='any', thresh=None,
                  subset=None, inplace=False)
```

- axis 默认axis=0/'index'表示按行删除；axis=1/'columns'则表示按列删除
- how 默认表示存在任意一个缺失值的行 (列)；how='all'表示整行 (列) 都是缺失值的行 (列)
- thresh 整数型数据，表示当行（列）中非空数据多于这个数值时，该行（列）被保留（threshold）
- subset 数组类型，表示从 subset 指定的行 (列) 中寻找缺失值
- inplace 默认筛选后的数据存为副本；若为True表示直接在原数据上更改

缺失值——补全

补全缺失值的主要方法：

1. 均值/中位数/众数补全：根据属性值的类型，用该属性取值的平均数（mean）/中位数（median）/众数（mode）进行补全
2. 固定值补全：将缺失的属性值用一个常量进行填补，如 0
3. 临近值补全：使用缺失值临近的值进行填补
4. 回归方法：对有缺失值的变量，根据已有数据和与其有关的其他变量的数据建立拟合模型来预测缺失的属性值

缺失值补全

Pandas 中fillna()函数可用来补全缺失值

```
df.fillna(value=None, method=None, axis=None,
          inplace=False, limit=None)
```

- value 指定用来填充的值，可以是单个值、字典类型（dict）或者其他的 DataFrame 对象
- method 插值填入的方式，默认使用value参数指定的数值填充；'backfill'/'bfill'表示用下一个非缺失值填充该缺失值；'pad'/'ffill'表示用前一个非缺失值去填充该缺失值
- axis 需要填充的轴，默认按列填充；axis=1/'index'表示按列填充
- limit 前向或后向填充时最大的填充范围, 默认全部填充。

固定值填充

可采用字典结构{key=value}定义所需填充的值，在用fillna()函数进行快速填充

```
values={'课程' : '金融大数据处理', '成绩' : 60}
df1 = df.fillna(value=values)
print(df1)
```

临近值填充

用缺失值所在列的前一个或后一个非缺失值填充缺失值填充

```
# forward fill, axis=0
    df1 = df.fillna(method='ffill')
    print(df1)
# backward fill, axis=0
    df2 = df.fillna(method='bfill')
    print(df2)
```

均值/中位数/众数/最大/最小值补充

实际数据分析工作中，常用平均值、中位数、最大值、最小值、众位数等来填充空值

```
df1 = df.fillna(df['成绩'].mean()) # 均值填充
df2 = df.fillna(df['成绩'].max()) # 最大值填充
df3 = df.fillna(df['成绩'].min()) # 最小值填充
df4 = df.fillna(df['成绩'].median()) # 中位数填充
df5 = df.fillna(df['成绩'].value_counts().index[0]) # 最高频次数值(众数)填充
```

数据格式规范问题

在实际数据处理工作中，数据的来源多种多样，数据格式可能各不相同。常见的格式不规范问题有：

1. 不同数据源对同一事物描述的单位不一致
2. 同类型数据，格式不一致
3. 数据格式不正确
4. 空白字符或特殊字符
5. 大小写不规范

数据格式查看

Pandas 中的数据类型和 Python 本身自带数据类型有所区别

类型	Pandas dtype	Python type
字符（文本）型数据	object	str
数值型整数类型	int64	int
浮点型数字类型	float64	float
布尔类型（True/False）	bool	bool
日期和时间类型	datetime64	NA

- `df.info()`：输出df对象中各列的索引、非空计数、数据类型以及内存使用。
- `df.dtypes()`：返回df对象中每一列的数据类型。

数据类型转换

Pandas 中astype('数据类型')函数可对列数据进行强制类型转换

```
df.astype(dtype, copy=True)
```

- dtype 需要被转换成的类型，可以选择'object'、'int64'、'float64'、'datetime64' 等；也可以使用字典方式{col1:dtype1, col2:dtype2, ...}，同时选择多个列进行不同类型的转换
- copy 是否将转换后的数据覆盖原数据，默认返回一个副本；设置copy=False则直接覆盖原数据

转换为日期型格式

Pandas 中to_datetime()函数可将指定数据转换成日期格式

```
df.to_datetime(arg, format=None)
```

- arg：需要处理的数据的列索引值
- format：日期格式，包括：
  - '%Y-%m-%d' 表示年月日格式
  - '%Y-%m-%d %H:%M:%S' 表示年月日并指定时分秒

转换为数值格式

Pandas 中`to_numeric()`函数可以把列数据转换成数值型的 float64 或 int64

```
df.to_numeric(arg, downcast)
```

- `arg`：需要处理的数据的列索引值
- `downcast`：转换后的数据类型，包括`'int64'`、`'object'`、`'float64'`（默认）

运行错误的问题

- 带有单位的数据，要删除单位才能转换成数值型
- 产生的不合理数据

空白字符的处理

空格在文本型数据中是非常普遍的存在，为了便于后续数据分析时能够进行统一计算，需要把无必要的空格进行删除。主要有：空格、制表符 (\t)、换行符 (\n)

空格可能的情况有：

- 文本字符中的空格。
- 数据内容为空白字符。

空白字符的检测

values()函数可将数据信息直观的显示出来。

```
print(df.values)
```

- \n：有换行符
- ' '：只有空白字符
- \t：有制表符

去除字符串前后的空格字符

Pandas 中去除字符型数据中的空白字符可以使用以下三个函数：

- strip 同时去除字符串左右两边的空格字符
- lstrip 去除字符串开头 (left) 的空格字符
- rstrip 去除字符串末尾 (right) 的空格字符

举例：

```
s = pd.Series(['1. Ant. ', '2. Bee!\n', '3. Cat?\t'])
print(s.str.lstrip('123.')) # 删除s序列中的左侧字符
print(s.str.strip()) # 删除s序列中各字符前后的空格
```

空白字符替换

对于字符间的空白符号可以用空字符进行替换，从而消除空白字符。Pandas 中的 replace()函数常被用来查找并替换数据

```
df.replace(to_replace=None, value=None, inplace=False,
           limit=None, regex=False, method='ffill')
```

- to_replace 表示需要被替换的值
- value 替换后的值
- inplace 是否要改变原数据表格，默认为不改变
- limit 限制替换的次数
- regex 是否使用正则表达式 (regular expression)，默认不使用
- method 替换方式，包括 pad/ffill（用前一个值去替换该值）、bfill (用后一个值去替换该值)

使用正则表达式替换空白字符

正则表达式'\s+'表示任意一个空白 (space) 字符

```
df = df.replace(r'\s+', '', regex=True)
```

注意：

正则表达式的前面要加上r，相应的匹配规则要放在单引号' '内。

此处r'\s+'是 Python 中表示任意空白字符的正则表达式

完全空白字符的替换

前后无字符的完全空字符的正则表达式为'^\s*$'，用该正则表达式转换为空值（NaN）

```
import numpy as np
df = df.replace(r'^\s*$', np.nan, regex=True)
```

说明:

^\s*$表示从字符串的开头 (^) 到结尾 ($) 都是空白字符 (\s)，*表示一个或多个（空白字符）

此外，还可以按照需要，将空白字符直接替换成一个特定值

```
df = df.replace(r'^\s*$', 0, regex=True)
```

数据格式统一

当从多个数据源采集数据时，不可避免的会出现数据格式不统一的问题，包括：

- 大小写统一：当某一列的数据为字母时，需要统一大小写格式。
- 单位统一：对某列同一件事物的描述，不同来源的数据可能会采用不同的单位。
- 小数位数统一：同一指标下的数值型数据，小数点后保留的位数需要进行统一化处理。
- 描述方式统一：同一种事物有多种不同的描述方式，比如性别列“男”和“M”，也应当统一。

大小写统一

Python 中用可以使用upper()、lower()、title()函数来统一大小写

函数	用途
str.upper()	将文本转换成全大写形式
str.lower()	将文本转换成全小写形式
str.title()	将文本转换成首字母大写，其余小写的形式

单位统一

利用 Pandas 中 replace() 函数进行单位的统一和单位字符的去除

```
# 去掉','和'$'两个字符，regex=True表示使用正则替换，
# 在字符串中匹配指定内容进行替换
df['信用额度'] = df['信用额度'].replace(',', '', regex=True)
                .replace('$', '', regex=True)
#统一万元和元的单位，去掉'元'
df['信用额度'] = df['信用额度'].replace('万', '0000',
                 regex=True).replace('元', '', regex=True)
#转换成float64数据类型
df['信用额度'] = df['信用额度'].astype('float64')
```

小数位数的统一

使用 Pandas 的round()函数

```
import pandas as pd
import numpy as np
df_round = pd.DataFrame(np.random.random([3, 3]),
                        columns=['A', 'B', 'C'],
                        index=['one', 'two', 'three'])
df = df_round*10
print(df)
print(df.round(0)) # 取整
print(df.round(2)) # 取两位小数
```

描述方式的统一

使用replace()进行字符串的替换

```
# 先转换为大写，然后再用映射关系，
#    将M转换为男，将F转换为女
df['性别'] = df['性别'].str.upper()
             .replace({'M' : '男', 'F' : '女'})
```

异常值

异常值通常指样本中数值明显偏离其余的观测值的个别值，也称为离群点（outlier)。异常值的分析也称为离群点分析。

包含:

- 简单统计量分析
- 箱型图分析
- 3 倍标准差

简单统计量分析

当数据量较大时，可以先对变量做描述性统计，Pandas 中的`describe()`函数可以展示列数据的统计性描述信息，包括：均值、方差、最大最小值、非空值数量、重复频次、不重复值个数等。

统计量描述函数

```
describe(percentiles=None, include=None, exclude=None,
         datetime_is_numeric=False)
```

- percentiles 设置百分位数统计量，默认为[.25,.5,.75]，表示数值从小到大排序后，位于 25%、50%、75% 数据量时的数值
- include 计算并输出哪类数据的统计量，默认值计算数值型列数据的特征统计量
- exclude 与include相反，哪些数据不需要计算特征统计量，默认不排除任何数据
- datetime_is_numeric 是否把时间类型当成数值型类型处理，默认不处理

描述函数的输出结果

数值型数据：

- count 非空数据的个数
- mean 数据平均值
- std 数据的标准差 (standard deviation)
- 25%/50%/75%：25%/50%/75% 分位上的数值
- max/min 数据的最大值/最小值

字符型数据：

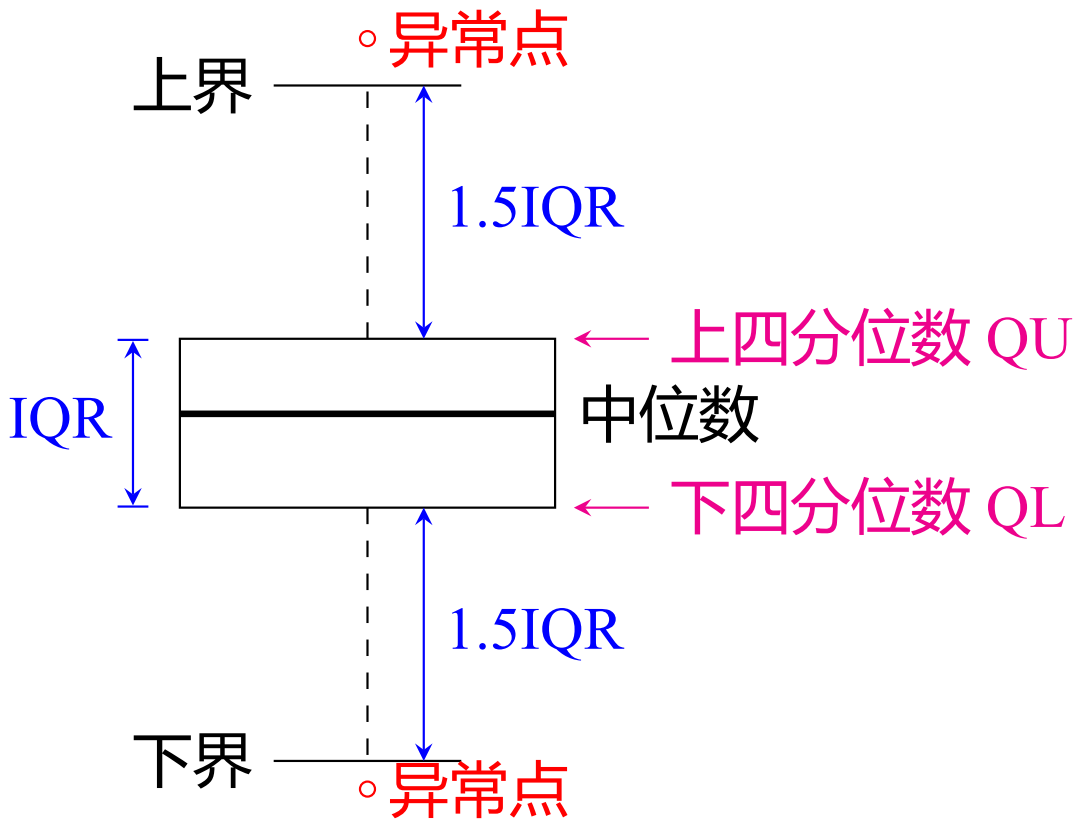
- count 非空数据的个数
- unique 去重后值的个数
- top 出现频次最高的值
- freq 出现的最高频次

字符型数据统计

```
# 指定输出文本类型列的统计信息
print(df.describe(include=['object']))
```

箱型图分析

箱型图 (box plot) 是一种用于显示一组数据分散情况的统计图，因形状如箱子而得名。它能显示出一组数据的最大值、最小值、中位数 (median)、及上下四分位数 (quartile)。



上下边界计算

```
des = df.describe()
q1 = des['工资']['25%']
q3 = des['工资']['75%']
iqr = q3-q1
min_value = q1-1.5*iqr # 下界限
max_value = q3+1.5*iqr # 上界限
print('分位差为:%.3f, 下限为:%.3f, 上限为:%.3f'
      %(iqr,min_value,max_value))
```

说明:

%.3f表示调整格式为保留小数点后 3 位的浮点型（float）数值。

绘制箱型图判断异常值

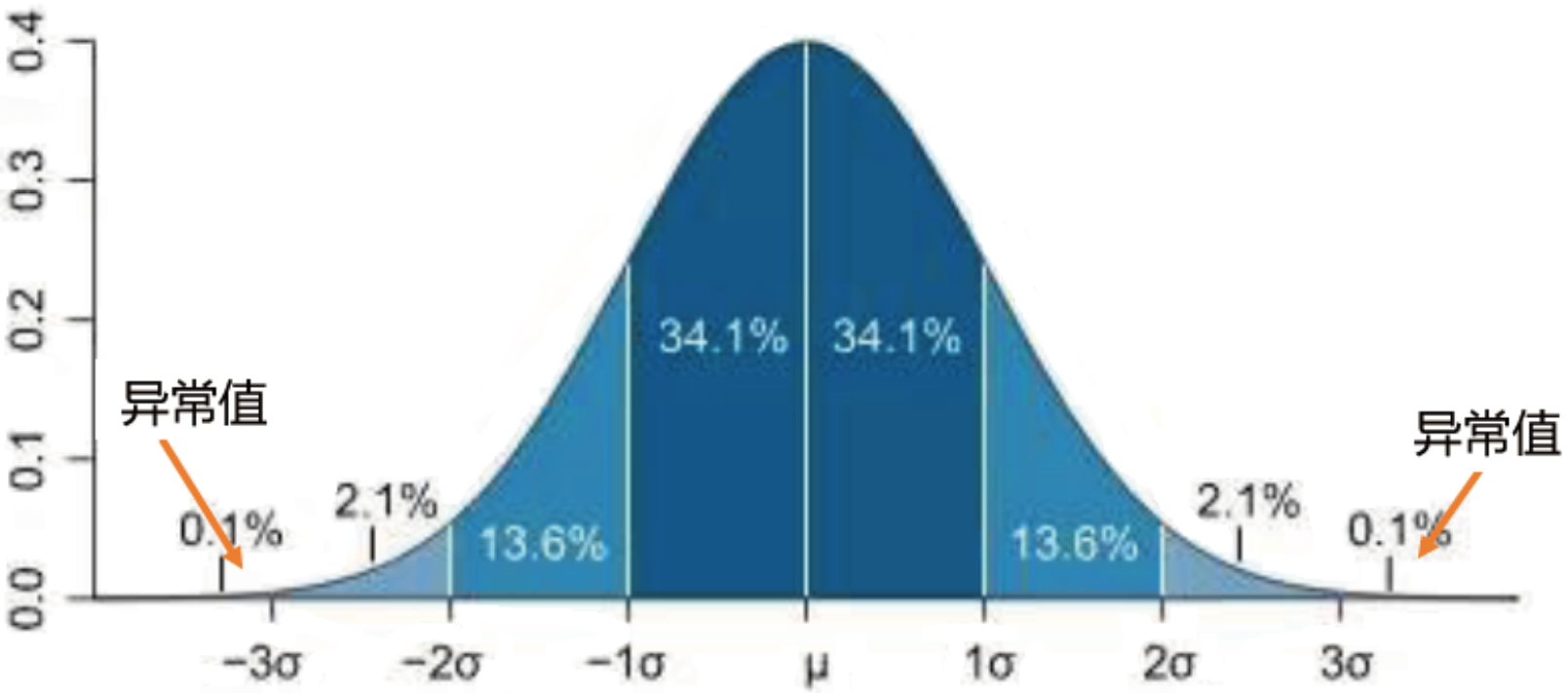
用 Pandas 中boxplot()函数直接绘制箱型图，可更加直观的观测异常值

```
df.boxplot(column=None, fontsize=None, rot=0,
           grid=True, figsize=None, return_type=None)
```

- column 指定要进行箱型图分析的列，输入为字符型或由字符型构成的列表
- fontsize 箱型图坐标轴字体大小
- rot 箱型图坐标轴旋转角度（rotation）
- grid 箱型图网格线是否显示，默认显示
- figsize 箱型图窗口尺寸大小
- return_type 指定返回对象的类型，默认不指定

3σ 原则

如果数据服从正态分布，在 3σ 原则下，异常值被定义为一组测定值中与平均值的偏差超过 3 倍标准差的值。



3σ 原则 (cont.)

举例:

```
des = df.describe()
std = des['工资']['std'] #获取“工资”列的标准差
mean = des['工资']['mean'] #获取“工资”列的平均值
data = df['工资']          #抽取df中“工资”列数据
print(data[abs(data-mean)>3*std])
```

异常值处理

- 删除异常值记录：直接将含有异常值的记录删除
- 视为缺失值：将异常值视为缺失值，按照缺失值处理的办法进行处理
- 平均值修正：可用前后两个观测值的平均值修正该异常值
- 不处理：直接在具有异常值的数据集上进行挖掘建模

异常值处理 (cont.)

使用.loc[]函数选定异常值所在的位置，进行重新赋值修改处理

```
df.loc[行索引名/[行索引列表], 列索引名/[列索引列表]] = value
```

举例:

```
# 选择df数据中，姓名是叶炜浩的行，“工号”列的数据，
# 并将该数据重新赋值为200107
    df.loc[df['姓名'=='叶炜浩', '工号']] = 200107
```

重复值的识别

Pandas 中的duplicated()函数通过返回布尔值序列，来判断 DataFrame 对象中，每一行是否与之前出现过的行相同，即是否存在重复的情况

```
duplicated(subset=None, keep='first')
```

- subset 指定进行重复值比较的列，默认对全部列进行比较
- keep 指定重复行的位置，并返回布尔值 True/False，选项有：{'first', 'last', False}。如果想标记出所有出现重复的行，则可设置keep=False

重复值的删除

Pandas 中的drop_duplicates()函数可用于删除duplicated()函数返回结果中被标记为True的行

```
df.drop_duplicates(subset=None, keep='first', inplace=False)
```

- subset 指定进行重复值比较的列，默认所有列
- keep 指定多个重复数据中保留行的位置，可以采用的选项有：{'first', 'last', False}
- 是否重新生成一个副本数据，默认表示直接在原数据上修改，True表示重新生成一个副本。

重复值的删除举例

```
print(df.drop_duplicates(subset=['姓名', '出生日期'],
                         keep='last'))
```

根据姓名和出生日期，查找并删除重复数据，保留其中的最后一条数据记录。

文本分析处理的概念

文本数据处理从非结构化文本数据中提取出有价值、易存储、结构化的数据。

常用的技术有：自然语言处理技术（NLP)、文字识别技术（OCR）

自然语言处理技术

自然语言处理，简称 NLP（Natural Language Processing）是以语言为对象，利用计算机技术来分析、理解和处理自然语言的一门学科

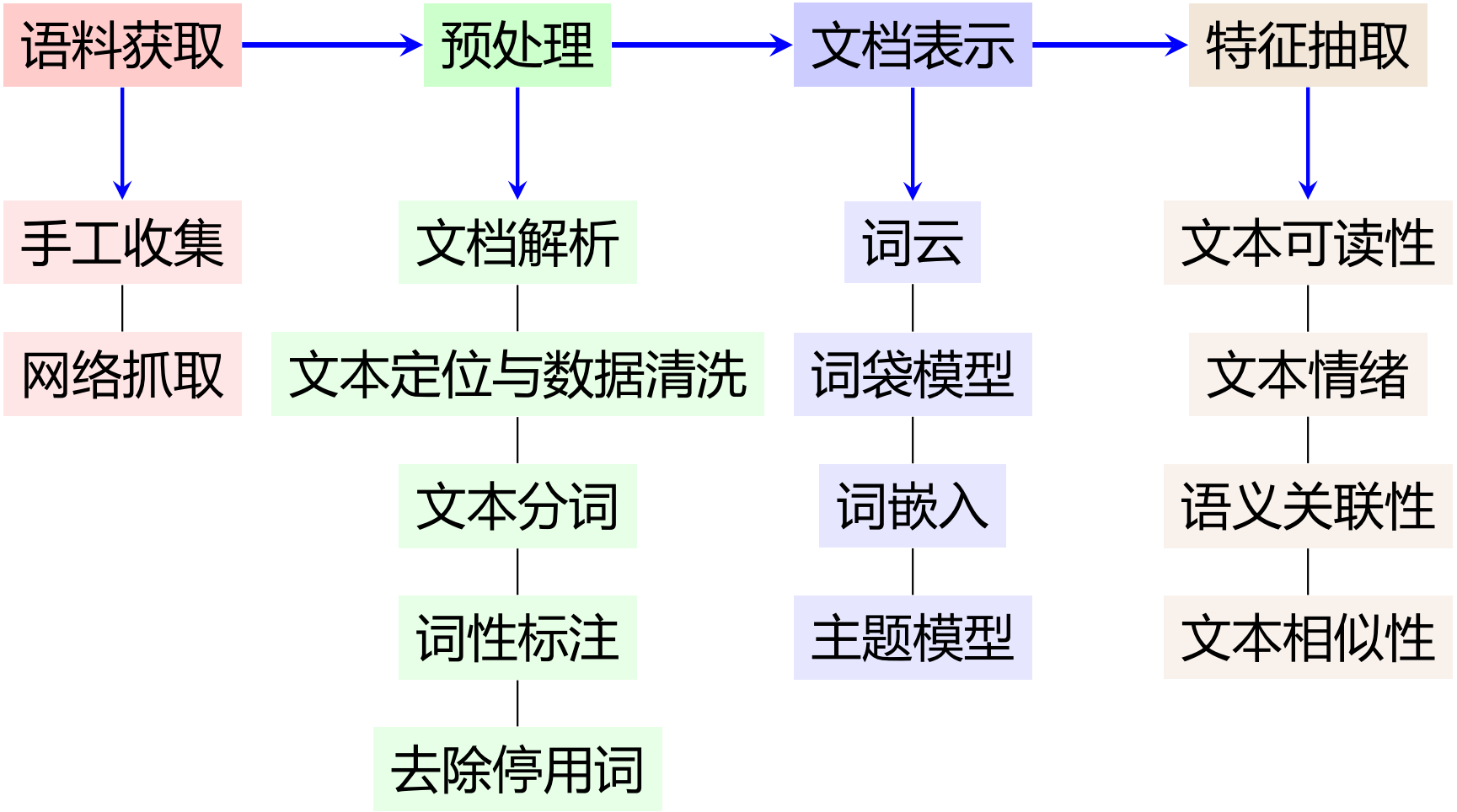
自然语言处理技术主要应用于机器翻译、舆情监测、自动摘要、观点提取、文本分类、自动问答、文本语义对比、语音识别等方面

文字识别技术

文字识别技术，也称光学字符识别（Optical Character Recognition，OCR），采用光学的方式将纸质文档中的文字转换成为黑白点阵的图像文件，并通过识别软件将图像中的文字转换成文本格式，供文字处理软件进一步编辑加工的技术

OCR 技术的应用场景：快递收件和寄件信息的填写、拍照识物、图片转文字、卡号识别、支票识别、财务报表信息识别等

文本分析处理的过程


语料获取
预处理
文档表示
特征抽取
手工收集
网络抓取
文档解析
文本定位与数据清洗
文本分词
词性标注
去除停用词
词云
词袋模型
词嵌入
主题模型
文本可读性
文本情绪
语义关联性
文本相似性

语料获取

- 手工收集：消耗大量的时间和人力成本
- 机器抓取：能够便捷、迅速地获取文本信息；抓取的同时对文本数据进行格式和内容的自动化、人工智能整理

预处理：文档解析

文档解析是进行文本预处理的第一步。在计算机领域，电子化文档被统称为富格式文档（Rich Text Document），这些文档包含文本段落、表格、图表等多种内容形态

解析富格式文档是进行文本预处理的第一步，即获取里面的信息内容

预处理：文本定位与数据清洗

文本定位是指对需要的文本信息进行定位，进而将该内容提取出来

数据清洗是指对文本中视为噪音的内容进行清洗和删除，噪音内容主要包括广告、超文本标记语言（HTML）、直译式脚本语言、图片等

预处理：文本分词

文本分词就是将句子、段落，分解为以字词为单位的结构。常用的分词方法有基于规则的和基于统计的

中文文本分词的注意事项

1. 切分颗粒度太小，容易破坏词语的意思
2. 针对歧义词，应该选择合适的分词模式
3. 信息爆炸的时代，新词汇层出不穷，及时的积累和快速的识别新词是一大难点

常用的分词工具有Jieba、HanLPL等。

词性标注

词性标注就是对切分后词语的词性做标记

通过词性标注，计算机能够识别词语的种类，消除词语歧义，进而能识别语法结构，降低计算机语义分析的难度

中英文在词性标注方面具有较大的差异

- 英文单词能通过词尾变换来揭示词性的变化
- 中文主要靠语法和语义来识别词性

去除停用词

停用词对句子语法结构很重要，但本身传达意义较少

- 英文：冠词（the, a）、连词（and, or）以及动词“to be”等
- 中文里的标点符号、特殊符号、表示逻辑关系的连接词（和、然而、因为、）以及俚语等

停用词增加了文本数据处理的难度，提高了文本分析的成本。

停用词的去除需要根据具体语义内容和分析目的来决定。

词云

词云技术能够描述词语在文本中出现的频率，当词语出现频率较高时，会以较大且醒目的形式呈现。

词袋模型

词袋模型（bag of words，BOW）是一种建立在文字词组语序不重要的假设之上，将文本看作是若干个词语的集合，只计算每个词语出现次数的一种文本向量化的表示方法

词嵌入

词嵌入是一种词的数值化表达，是将词汇映射到一个多维向量空间中的实数向量的方法统称

通过计算词向量之间夹角余弦值（cosine）而得出单词之间的相似性

主题模型

主题模型会自动分析文档，统计词语信息，断定当前文档含有哪些主题，以及每个主题所占的比例。如 LDA（Latent Dirichlet Allocation, 隐狄利克雷分配）模型。

特征抽取

- 文本可读性：反映了读者理解文本信息的难易程度。
- 文本情绪：提取方法主要包括词典法和有监督机器学习方法。
- 语义关联性：根据某一类词语去识别文本语义特征的过程。
- 文本相似性：文本数据处理中，经常需要判断两个文本是否相似，并计算相似程度。