

数据分析实战

讲师名称：李川

Can//ay 嘉为数字咨询

数 字 化 人 才 培 养 先 行 者



目录

1

分词简介

2

jieba 的分
词模式

3

分词流程

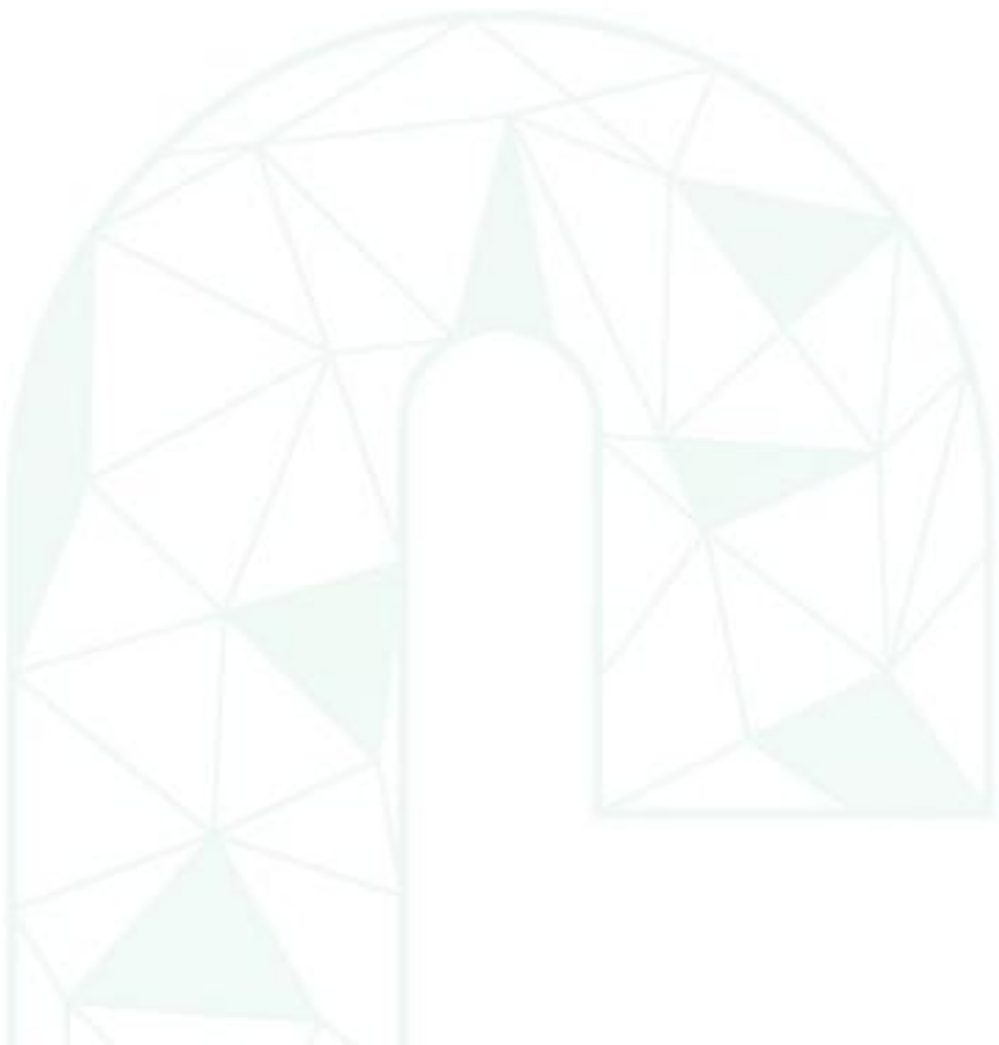
4

数据合并

5

自动化报告

CONTENTS



/01

分词简介

1 分词

文本预处理中的“分词”是将连续的文本（如句子、段落）拆分为有意义的最小语言单位（如词语、子词）的过程，是自然语言处理（NLP）的基础步骤。不同语言的分词逻辑不同（中文无天然分隔符，需特殊处理），以下结合中文场景详细介绍分词方法与工具。

1.1 中文分词核心问题

中文文本没有像英文那样的空格分隔符，且存在歧义（如“乒乓球拍卖完了”可拆分为“乒乓球 / 拍卖 / 完了”或“乒乓球拍 / 卖 / 完了”），因此需要通过词典匹配、统计模型等方式实现准确分词。

1.2 分词工具对比

工具	特点	适用场景
Jieba	轻量、速度快，支持自定义词典，适合中文常规场景	博客、新闻、通用文本分词
THULAC	清华大学开发，分词准确率高，支持词性标注	学术研究、高精度需求场景
SnowNLP	针对中文优化的 NLP 工具包，集成分词、情感分析等	简单中文文本处理流水线
HanLP	支持多语言，分词能力强，适合复杂场景（如命名实体识别）	企业级应用、专业领域（如法律、医疗）



/02

jieba 的分词模式

2 jieba 的分词模式

jieba 是中文分词领域最常用的工具之一，其核心优势在于支持多种分词模式，可根据不同场景（如精准分析、快速检索、新词挖掘等）灵活选择。jieba 的分词模式主要分为 3 种基础模式（精准模式，全模型，搜索模式），以及衍生的 自定义分词能力

2.1 精确模式（默认模式）

核心逻辑：试图将文本最精确地切分，不存在冗余词语，适合需要“准确拆分语义单元”的场景。

特点：拆分结果最贴合常规语义，无重复或多余词语，是最常用的模式。

适用场景：文本分析（如情感分析、关键词提取）、自然语言理解（如问答系统）等。

```
import jieba

text = "我喜欢自然语言处理和机器学习"
# 精确模式（默认，无需额外参数）
result = jieba.lcut(text) # lcut 返回列表，cut 返回生成器
print(result)

#输出：['我', '喜欢', '自然语言处理', '和', '机器学习']
```

2.2 全模式

核心逻辑：将文本中所有可能的词语都扫描出来，存在冗余词语，适合需要“穷尽所有可能词语”的场景。

特点：拆分结果包含所有词典中匹配到的词语（可能重复或交叉），不考虑语义连贯性。

适用场景：关键词检索（如搜索引擎需匹配所有可能相关的词）、新词发现（辅助挖掘潜在词语组合）等。

```
# 全模式：参数 cut_all=True
result = jieba.lcut(text, cut_all=True)
print(result)
#['我', '喜欢', '自然', '自然语言', '自然语言处理', '语言', '处理', '和', '机器', '机器学习', '学习']
```

2.3 搜索引擎模式

核心逻辑：在精确模式的基础上，对长词进一步拆分（补充细分词语），兼顾精确性和全面性。

特点：既保留精确模式的核心语义，又拆分长词为更短的子词，适合“需要兼顾精准和召回”的场景。

适用场景：搜索引擎分词（如用户搜索“自然语言处理”时，同时匹配“自然语言”“处理”等子词）、信息检索等。

```
# 搜索引擎模式：使用 jieba.lcut_for_search() 方法
```

```
result = jieba.lcut_for_search(text)
```

```
print(result)
```

```
#输出：['我', '喜欢', '自然', '语言', '自然语言', '处理', '自然语言处理', '和', '机器', '学习',  
'机器学习']
```

2.4 jieba 的衍生分词能力（增强模式）

除基础模式外，jieba 还支持通过自定义配置增强分词效果，可视为 “模式的扩展”

2.4.1 自定义词典（解决专业术语拆分问题）

核心需求：默认词典可能缺少专业领域词汇（如 “大语言模型” “Transformer”）、网络新词（如 “内卷” “躺平”），导致拆分错误（如 “大语言模型” 被拆为 “大 / 语言 / 模型”）。

使用方法：通过 `jieba.load_userdict()` 加载自定义词典（格式：词语 词频 词性，词频和词性可选），或用 `jieba.add_word()` 临时添加词语。

```
# 未添加自定义词时
text = "大语言模型是AI的热点"
print(jieba.lcut(text)) # 输出：['大', '语言', '模型', '是', 'AI', '的', '热点']

# 添加自定义词后
jieba.add_word("大语言模型") # 临时添加
print(jieba.lcut(text)) # 输出：['大语言模型', '是', 'AI', '的', '热点']
```

2.4.2 停用词过滤（优化分词结果）

核心需求：分词结果中可能包含无实际语义的词语（如 “的” “是” “在” ），需过滤以减少噪声。

使用方法：自定义停用词表（如 stopwords = {"的", "是", "在"}），分词后过滤掉包含在停用词表中的词语。

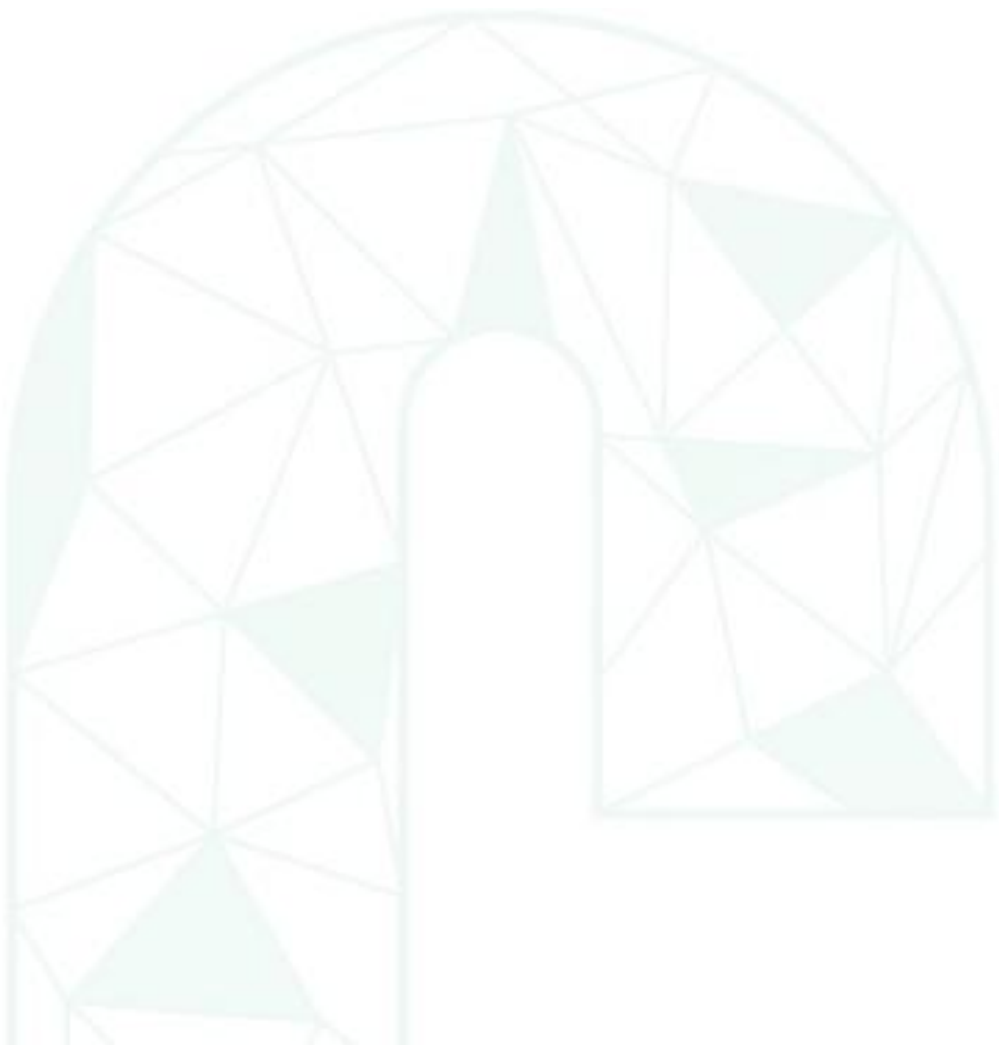
```
text = "今天的天气真好，我打算出去散步"
words = jieba.lcut(text)
stopwords = {"的", " ", "我"} # 自定义停用词
filtered_words = [word for word in words if word not in stopwords]
print(filtered_words) # 输出：['今天', '天气', '真好', '打算', '出去', '散步']
```

2.4.3 词性标注（结合词语属性的分词）

核心需求：不仅需要拆分词语，还需知道词语的词性（如名词、动词），用于后续语法分析、语义理解。

使用方法：通过 jieba.posseg 模块实现，返回 “词语 + 词性” 的元组。

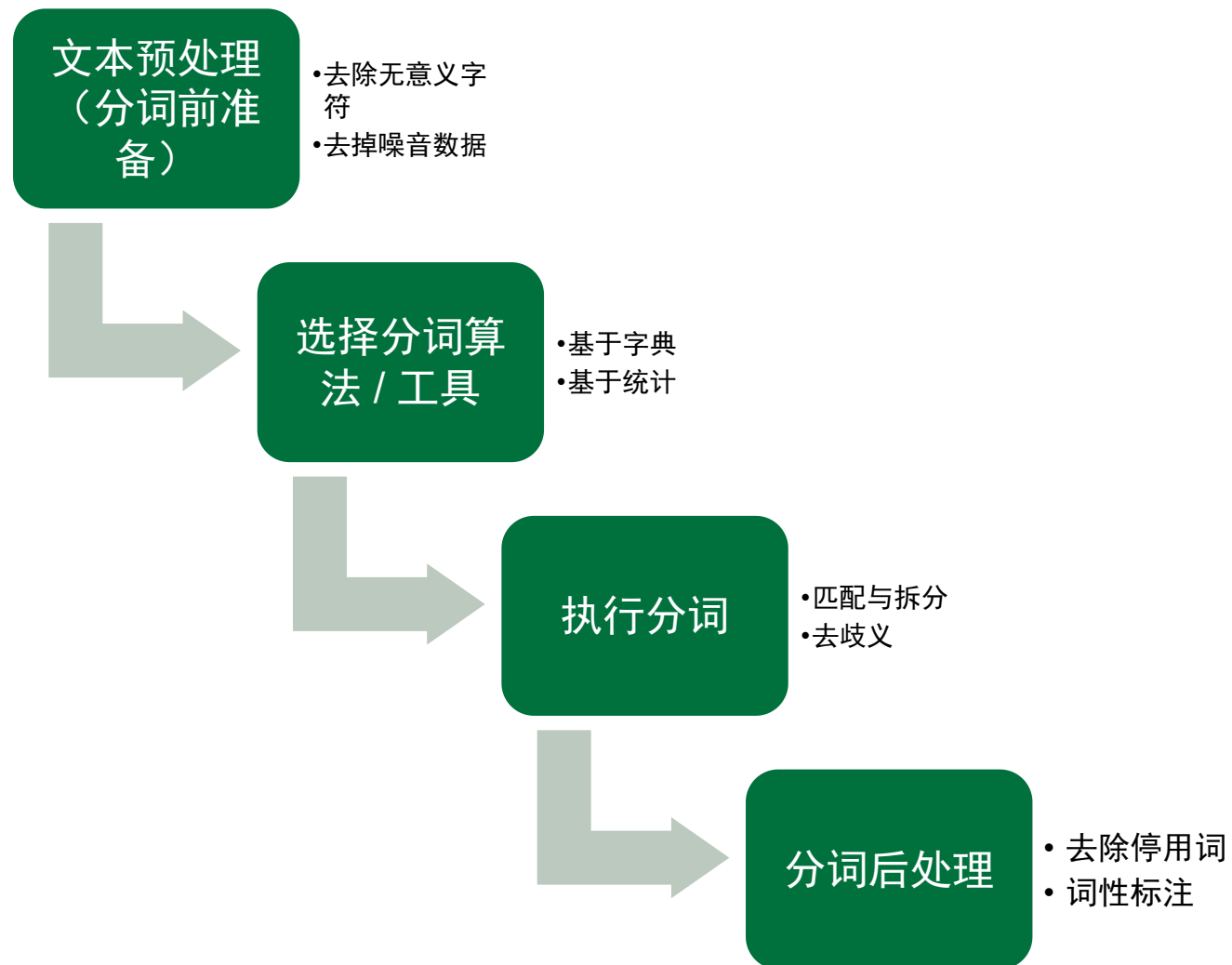
```
import jieba.posseg as pseg
text = "小明在图书馆看书"
words_with_pos = pseg.lcut(text) # 返回（词语，词性）列表
print([(word, pos) for word, pos in words_with_pos])
#输出：[('小明', 'nr'), ('在', 'p'), ('图书馆', 'n'), ('看', 'v'), ('书', 'n')]
（词性说明：nr= 人名，p= 介词，n= 名词，v= 动词）
```



/03

分词流程

3.1 通用分词流程



3.2 通用分词流程（中文）

1. 文本预处理（分词前准备）

在拆分前需先处理文本中的干扰信息，避免影响分词准确性，常见操作包括：

去除无意义字符：删除标点符号（如 “，。！”）、特殊符号（如 “@#¥”）、空格、换行符等；

统一文本格式：例如大小写转换（中文无大小写，但英文需统一）、全半角转换（如 “，” → “,”）；

过滤噪声数据：如去除超短文本（仅含符号）、替换敏感词（如需）。

示例：

原始文本：“Hello! 今天天气真好，适合出去玩~”

预处理后：“Hello 今天天气真好 适合出去玩”

3.2 通用分词流程（中文）

2. 选择分词算法 / 工具

根据语言特点和需求选择分词方法，中文分词常用算法如下（工具如 jieba、THULAC 等已封装这些算法）：

基于词典匹配（最基础）：

核心：用预设词典（如《现代汉语词典》）匹配文本中的词语，按“最长匹配”“最短匹配”等规则拆分。

例：词典含“自然”“自然语言”，则“自然语言处理”优先匹配最长的“自然语言”。

基于统计模型（更智能）：

核心：通过机器学习（如隐马尔可夫模型 HMM、条件随机场 CRF）学习文本中词语的出现概率，预测最优拆分方式（即使词典中无该词，也能根据上下文拆分）。

例：“人工智能”未在词典中，但模型可通过“人工”“智能”的常见搭配拆分。

混合策略（主流工具采用）：

如 jieba 结合“词典匹配 + 统计模型”，兼顾效率和准确性。

3.2 通用分词流程（中文）

3. 执行分词（核心拆分步骤）

调用分词工具，根据算法对预处理后的文本进行拆分，以 jieba 分词 为例，包含以下细分逻辑：

加载词典：默认加载内置词典（含常用词），支持用户添加自定义词典（如专业术语：“大语言模型”）；

匹配与拆分：

先通过词典匹配已知词，未匹配到的部分用统计模型预测拆分（如网络新词 “yyds” “绝绝子”）；

处理歧义：

对多义词或歧义结构（如 “乒乓球拍卖完了” 可拆分为 “乒乓球 / 拍卖 / 完了” 或 “乒乓球拍 / 卖 / 完了”），通过上下文概率判断最优拆分。例如：

```
import jieba
text = "自然语言处理是人工智能的重要分支"
# 精确模式（默认）：最常用，适合文本分析
words = jieba.lcut(text) # lcut 返回列表
print(words)
# 输出：['自然语言处理', '是', '人工智能', '的', '重要', '分支']
```

3.2 通用分词流程（中文）

4. 分词后处理（优化结果）

拆分后可能存在冗余或错误结果，需进一步优化：

去除停用词：删除无实际语义的高频词（如 “的” “是” “在” ），需结合停用词表（可自定义）；

```
stopwords = {"的", "是"} # 自定义停用词表
```

```
filtered_words = [word for word in words if word not in stopwords]
```

```
print(filtered_words) # 输出：['自然语言处理', '人工智能', '重要', '分支']
```

3.2 通用分词流程（中文）

4. 分词后处理（优化结果）

修正错误拆分：对明显错误的结果（如专业术语被拆错），通过添加自定义词典解决；例如：

```
jieba.add_word("大语言模型")
```

```
# 添加自定义词
```

```
text = "大语言模型是NLP的热点"
```

```
print(jieba.lcut(text))
```

```
# 输出：['大语言模型', '是', 'NLP', '的', '热点']
```

3.2 通用分词流程（中文）

4. 分词后处理（优化结果）

词性标注（可选）：部分工具支持给分词结果标注词性（如“名词”“动词”），为后续任务提供信息。例如：

```
import jieba.posseg as pseg
words_with_pos = pseg.lcut("我爱吃苹果")
print([(word, pos) for word, pos in words_with_pos])
# 输出：[('我', 'r'), ('爱', 'v'), ('吃', 'v'), ('苹果', 'n')]
# （r：代词，v：动词，n：名词）
```

3.2 通用分词流程（中文）

4. 分词后处理（优化结果）

关键字提取（可选）：

- `jieba.analyse.extract_tags()` 是 jieba 库中用于关键词提取的核心函数，基于 TF-IDF（词频 - 逆文档频率）算法从文本中提取重要程度最高的词语，**核心原理**：TF-IDF 算法通过两个维度衡量词语重要性：
 - **TF（词频）**：词语在当前文本中出现的频率（频率越高越重要）；
 - **IDF（逆文档频率）**：词语在所有文档中出现的频率（在少数文档中出现的词更具区分度，IDF 更高）。
 - 最终权重 = $TF \times IDF$ （权重越高，词语对文本的代表性越强）。

核心语法如下：

```
jieba.analyse.extract_tags(  
    sentence,          # 待提取关键词的文本（字符串）  
    topK=20,           # 提取的关键词数量（默认20个）  
    withWeight=False,  # 是否返回关键词权重（True返回元组列表，False返回词语列表）  
    allowPOS=(),        # 允许提取的词性（如('n','v')只提取名词和动词，默认所有词性）  
    withFlag=False     # 是否返回词性（True时返回(词语, 词性)，0.42+版本支持）  
)
```


4.3 英文分词流程（与中文的差异）

英文天然以空格分隔词语，基础分词更简单，但复杂场景仍需优化：

预处理：去除标点（如 “.” “,” ）、大小写转换（如 “Natural” → “natural” ）；

基础拆分：按空格和标点拆分（如 “Hello, world!” → “Hello” “world” ）；

进阶处理：

处理缩写（如 “don't” → “do” “not” ）、连字符（如 “state-of-the-art” → “state” “of” “the” “art” ）；

子词拆分（针对未登录词，如 “unhappiness” → “un” “happiness” ，通过工具如 NLTK、spaCy 实现）。

4.4 中文分词实例

例4-1核工业相关新闻片段分词

```
import jieba
import jieba.analyse
import jieba.posseg as pseg # 词性标注模块
from wordcloud import WordCloud
import matplotlib.pyplot as plt
text = """
核反应堆是核电站的核心设备，其安全运行直接关系到公众健康。
铀浓缩技术是核燃料生产的关键，需严格控制浓度以避免核扩散。
数字化监测系统提升了核设施的运行可靠性，保障核电安全。
"""

print("=== 1. 基础分词模式 ===")
# 精确模式（默认）：最常用，切分最准确，适合文本分析
exact_cut = jieba.lcut(text)
print("精确模式：", exact_cut)
# 全模式：把所有可能的词语都切分出来，可能有冗余
full_cut = jieba.lcut(text, cut_all=True)
print("全模式：", full_cut)
# 搜索引擎模式：在精确模式基础上，对长词进一步拆分（适合搜索引擎索引）
search_cut = jieba.lcut_for_search(text)
print("搜索引擎模式：", search_cut)
```

4.4 中文分词实例

例4-1核工业相关新闻片段分词

```
# -----
# 2. 自定义词典（解决专业术语分词问题）
# -----
print("\n=== 2. 自定义词典分词 ===")

# 问题：默认分词可能拆分专业术语（如“核反应堆”可能被拆分为“核/反应堆”）
# 解决：创建自定义词典（格式：词语 词频(可选) 词性(可选)）
with open("nuclear_dict.txt", "w", encoding="utf-8") as f:
    f.write("核反应堆 100 n\n") # n表示名词
    f.write("铀浓缩 80 n\n")
    f.write("核燃料 60 n\n")

# 加载自定义词典
jieba.load_userdict("nuclear_dict.txt")
custom_cut = jieba.lcut(text)
print("加载专业词典后：", custom_cut)
```

4.4 中文分词实例

例4-1核工业相关新闻片段分词

```
# -----  
# 3. 词性标注（标记词语类型）  
# -----  
print("\n=== 3. 词性标注 ===")  
# 切分并标记词性（n: 名词, v: 动词, a: 形容词等）  
words_with_pos = pseg.lcut(text)  
# 提取前5个词展示  
print("词语 + 词性: ", [(word, flag) for word, flag in words_with_pos[:5]])  
# -----  
# 4. 去除停用词（过滤无意义词汇）  
# -----  
print("\n=== 4. 去除停用词 ===")  
# 定义停用词表（无实际意义的词）  
stopwords = {"的", "是", "其", "和", "以", "直接", "到", "需"}  
# 先分词, 再过滤停用词  
cut_words = jieba.lcut(text)  
filtered_words = [word for word in cut_words if word not in stopwords]  
print("分词后（含停用词）: ", cut_words[:10]) # 前10个  
print("过滤后（去停用词）: ", filtered_words[:10]) # 前10个
```

4.4 中文分词实例

例4-1核工业相关新闻片段分词

```
# -----  
# 5. 关键词提取（基于TF-IDF）  
# -----  
print("\n=== 5. 关键词提取 ===")  
# 从文本中提取前3个关键词（带权重）  
keywords = jieba.analyse.extract_tags(  
    text,  
    topK=3,          # 提取数量  
    withWeight=True, # 返回权重  
    allowPOS=()      # 允许的词性（默认所有）  
)  
print("关键词（权重越高越重要）：")  
for word, weight in keywords:  
    print(f"{word}: {weight:.4f}")
```

4.5 生成词云图

词云图 (Word Cloud) 是一种将文本中出现频率较高的词语以视觉化方式呈现的图表——出现频率越高的词语，在词云中显示得越大、越突出，能直观展示文本的核心主题。

常用工具：Python 的 wordcloud 库（核心生成工具）+ jieba（中文分词）。

4.5 生成词云图实例

例4-2核工业相关新闻片段词云图

```
#例4-2 词云图
print("\n=== 6. 词云图 ===")
# 1. 将jieba分词并过滤停用词后的词拼接成词云所需的空格分隔格式
word_list = " ".join(filtered_words)
# 2. 生成词云（设置参数）
wc = WordCloud(
    font_path="simhei.ttf", # 中文字体（需确保本地有该字体，或替换为其他中文字体路径）
    background_color="white", # 背景色
    max_words=20, # 最多显示20个词
    width=800, height=400 # 尺寸
)
wordcloud = wc.generate(word_list) # 根据词语生成词云
# 3. 显示词云
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud)
plt.axis("off") # 隐藏坐标轴
plt.show()
```



/04

数据合并

5.1 concat()函数

`concat()` 函数用于沿着指定轴（行或列）拼接多个 DataFrame 或 Series，是数据合并的基础工具之一。它能灵活处理不同来源、不同结构的数据组合，常用于数据整合、批量处理后的结果汇总等场景。。

核心语法格式如下：

```
pd.concat(  
    objs,          # 要拼接的对象列表（如[df1, df2, df3]）  
    axis=0,        # 拼接轴：0=行方向（纵向），1=列方向（横向）  
    join='outer',  # 连接方式：'outer'取并集（默认），'inner'取交集  
    ignore_index=False # 是否重置索引（默认保留原索引）  
)
```

5.2 merge()函数

`merge()` 函数用于基于一个或多个键（key）将两个或多个 DataFrame 按行合并，类似于 SQL 中的表连接（JOIN）操作。它是处理关系型数据（如订单表与用户表、销售表与产品表）的核心工具，能精准关联不同数据源的信息。

核心语法格式如下：

```
pd.merge(  
    left,          # 左侧DataFrame  
    right,         # 右侧DataFrame  
    how='inner',   # 连接方式（inner/outer/left/right）  
    on=None,       # 连接键（左右表共有的列名）  
    left_on=None,  # 左侧表的连接键  
    right_on=None, # 右侧表的连接键  
    suffixes=('_x', '_y') # 列名重复时的后缀  
)
```

5.3 join()方法

DataFrame对象的join()方法也可实现主键合并的功能，但它主要是根据DataFrame的索引来进行合并的。语法格式如下：

```
DataFrame_obj.join(other, on=None, how="left", lsuffix="", rsuffix="", sort=False)
```

参数说明：

other：连接的另外一个数据对象，连接时是通过它的索引进行连接

on：指调用者要通过哪个列去进行连接

how：可选项有"inner"、"outer"、"left"、"right"，分别表示取交集、并集、取左表及其交集部分、右表及其交集部分，默认为"left"

lsuffix：当发生重复时左边数据对象设置后缀

rsuffix：当发生重复时为右边数据对象设置后缀

sort：是否按连接关键字排序，默认为False。

5.4 重叠合并

在处理数据的过程中，当一个DataFrame(Series)对象中出现了缺失数据，而对于这些缺失数据，我们希望能使用其他DataFrame(Series)对象中的数据填充，这时可以通过combine_first()方法填充缺失数据。combine_first()进行重叠合并时，两个数据对象的行索引和列索引必须有重叠的部分。

语法格式如下：

```
DataFrame_obj(Series_obj).combine_first(other)
```

参数说明：

other：用于填充的DataFrame或Series。

5.5 数据合并案例

例4-3 核工业燃料棒生产数据合并的应用案例

```
import pandas as pd
# -----
# 1. 准备数据（模拟不同来源的核燃料数据）
# -----
# 一车间生产数据
workshop1 = pd.DataFrame({
    '燃料棒ID': ['FR-001', 'FR-002', 'FR-003'],
    '铀浓度(%)': [4.5, 4.7, 4.6],
    '生产时间': ['2023-10-01', '2023-10-02', '2023-10-03'],
    '车间': '一车间'
})
# 二车间生产数据（结构与一车间相同）
workshop2 = pd.DataFrame({
    '燃料棒ID': ['FR-004', 'FR-005'],
    '铀浓度(%)': [5.0, 4.8],
    '生产时间': ['2023-10-01', '2023-10-03'],
    '车间': '二车间'
})
```

5.5 数据合并案例

例4-3 核工业燃料棒生产数据合并的应用案例

检测数据（需与生产数据关联）

```
inspection = pd.DataFrame({
    '燃料棒ID': ['FR-001', 'FR-002', 'FR-003', 'FR-004', 'FR-005'],
    '检测结果': ['合格', '合格', '待复检', '合格', '不合格'],
    '检测员': ['张工', '李工', '王工', '赵工', '孙工']
})
```

```
print("原始数据：")
```

```
print("一车间数据:\n", workshop1, "\n")
```

```
print("二车间数据:\n", workshop2, "\n")
```

```
print("检测数据:\n", inspection, "\n")
```

```
# -----
```

2. 使用concat()纵向合并生产数据

```
# -----
```

合并两个车间的生产记录（纵向拼接）

```
production_all = pd.concat([workshop1, workshop2], ignore_index=True)
```

```
print("2. 合并后的生产数据：")
```

```
print(production_all, "\n")
```

5.5 数据合并案例

例4-3 核工业燃料棒生产数据合并的应用案例

```
# -----  
# 3. 使用merge()关联生产数据与检测数据  
# -----  
# 基于"燃料棒ID"连接生产数据和检测数据（内连接）  
combined_data = pd.merge(  
    production_all,  
    inspection,  
    on='燃料棒ID', # 连接键  
    how='inner'    # 只保留双方都有的记录  
)  
  
print("3. 生产+检测合并数据：")  
print(combined_data)
```



/06

自动化报告

6.1 自动化报告

自动化报告 (Automated Reporting) 是通过工具或代码自动生成、格式化和分发报告的过程，核心目标是减少人工操作、确保格式规范、提升生成效率，尤其适用于需要定期输出（如日报 / 周报）、格式固定（如业务报表、实验报告）或数据驱动（如数据分析报告）的场景。

6.2 实现报告自动化的核心流程

1. 明确报告需求与模板设计

内容确定：需包含的固定模块（如标题、日期、负责人）、动态数据（如 KPI 指标、图表）、分析结论框架。

模板规范化：用工具（如 Word、Markdown、LaTeX）定义统一格式，标记“动态数据占位符”（如{{销售额}}）。

2. 数据对接与处理自动化

数据源连接：通过代码（Python/R）对接数据库（MySQL、SQL Server）、文件（Excel、CSV）、API 接口等。

自动计算与清洗：用脚本自动完成数据过滤、聚合、指标计算（如“月度环比增长率”），输出报告所需的结构化数据。

6.2 实现报告自动化的核心流程

3. 报告生成与格式化

工具选择：根据报告类型（文本 / 图表 / PDF）选择工具，如 Python 的 Jinja2（模板引擎）、Python-docx（Word）、ReportLab（PDF）、R Markdown（多格式）。

动态填充：将处理后的数据自动填充到模板的占位符中，生成完整报告。

4. 分发与通知自动化

自动将生成的报告保存到指定路径（本地 / 云存储），或通过邮件、企业微信 / 钉钉机器人推送给相关人员。

6.3 自动化报告实例

例4-4 核燃料生产周报告

```
import pandas as pd
import matplotlib.pyplot as plt
from docx import Document
from docx.shared import Inches
from datetime import datetime, timedelta
# -----
# 关键修复：设置Matplotlib中文字体（放在绘图前）
# -----
plt.rcParams["font.family"] = ["SimHei", "WenQuanYi Micro Hei", "Heiti TC"] # 支持中文
plt.rcParams["axes.unicode_minus"] = False # 解决负号显示问题（避免方块）
# 1. 模拟数据（生产数据）
dates = [datetime.now() - timedelta(days=i) for i in range(7, 0, -1)]
data = {
    "日期": [d.strftime("%Y-%m-%d") for d in dates],
    "生产数量(根)": [120, 135, 128, 140, 132, 145, 150],
    "合格率(%)": [98.2, 97.8, 98.5, 99.0, 98.7, 99.2, 99.5]
}
df = pd.DataFrame(data)
```

6.3 自动化报告实例

例4-4 核燃料生产周报告

```
# 2. 生成趋势图表（已支持中文）
plt.figure(figsize=(8, 4))

# 生产数量（柱状图）
plt.bar(df["日期"], df["生产数量(根)"], color="#4CAF50", alpha=0.7, label="生产数量")
plt.ylabel("生产数量(根)") # 中文标签

# 合格率（折线图，共享x轴）
ax2 = plt.twinx()
ax2.plot(df["日期"], df["合格率(%)"], color="#2196F3", marker="o", label="合格率")
ax2.set_ylabel("合格率(%)") # 中文标签

plt.title("核燃料棒周生产趋势") # 中文标题
plt.tight_layout() # 自动调整布局，避免标签重叠
plt.savefig("production_trend.png") # 保存图表（此时中文已正常显示）
```

6.3 自动化报告实例

```
# 3. 生成Word报告
doc = Document()
doc.add_heading("核燃料生产周报告", 0)
# 添加基本信息
doc.add_paragraph(f"报告周期: {df['日期'].min()} 至 {df['日期'].max()}")
doc.add_paragraph(f"生成时间: {datetime.now().strftime('%Y-%m-%d %H:%M')}")
# 添加汇总数据
doc.add_heading("一、周汇总指标", level=1)
table = doc.add_table(rows=2, cols=2)
table.cell(0, 0).text = "总生产数量"
table.cell(0, 1).text = f"{df['生产数量(根)'].sum()} 根"
table.cell(1, 0).text = "平均合格率"
table.cell(1, 1).text = f"{df['合格率(%)'].mean().round(2)}%"
# 添加图表
doc.add_heading("二、生产趋势分析", level=1)
doc.add_paragraph("本周生产数量与合格率均呈上升趋势, 合格率较上周提升0.5%。")
doc.add_picture("production_trend.png", width=Inches(6)) # 插入修复后的图表
# 保存报告
report_path = f"核燃料生产周报告_{df['日期'].max()}.docx"
doc.save(report_path)
print(f"报告生成成功: {report_path}")
```

The logo features a central green diamond containing the text. This diamond is surrounded by a larger, lighter green diamond. To the left of the central diamond is a dark green chevron shape pointing right. Several small diamonds in dark green, light green, and grey are scattered around the bottom and right sides of the main graphic. A thin horizontal line passes through the center of the composition.

Canllay 嘉为数字咨询

数字化人才培养
先行者