[ ]:
```
Day 10 / 数据可视化进阶
    词云图
        文本处理
            分词
            删除停止词
        词云生成
    分词的应用: 搜索引擎
    情感极性分析
    配色
        色板网站
        从图片生成色板
```

[6]:
```
!pip install jieba
```

```
Requirement already satisfied: jieba in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (0.42.1)
Collecting wordcloud
  Downloading wordcloud-1.9.2.tar.gz (222 kB)

222.8/222.8 kB 334.7 kB/s eta 0:00:00 kB/s eta
0:00:01:01
  Preparing metadata (setup.py) … done
Requirement already satisfied: numpy>=1.6.1 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
wordcloud) (1.24.2)
Requirement already satisfied: pillow in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
wordcloud) (9.5.0)
Requirement already satisfied: matplotlib in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
wordcloud) (3.7.1)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (4.39.3)
Requirement already satisfied: cycler>=0.10 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (2.8.2)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (1.0.7)
Requirement already satisfied: packaging>=20.0 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
matplotlib->wordcloud) (23.0)
Requirement already satisfied: six>=1.5 in
/Users/xiejiss/.pyenv/versions/3.10.10/lib/python3.10/site-packages (from
python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Installing collected packages: wordcloud
  DEPRECATION: wordcloud is being installed using the legacy 'setup.py
install' method, because it does not have a 'pyproject.toml' and the 'wheel'
package is not installed. pip 23.1 will enforce this behaviour change. A
possible replacement is to enable the '--use-pep517' option. Discussion can be
found at https://github.com/pypa/pip/issues/8559
  Running setup.py install for wordcloud … done
Successfully installed wordcloud-1.9.2

[notice] A new release of pip
available: 22.3.1 -> 23.2.1
[notice] To update, run:
pip install --upgrade pip
```

0.0.1 词云

词云（Word Cloud）是一种可视化方式，用于展示一段文本中频繁出现的词汇，以便于观察和分析。词云通过将文本中的词汇按照其出现频率或重要性进行可视化，将常见的词汇以较大的字体大小展示，而罕见的词汇则以较小的字体大小展示。这样，词云图形在一瞥之间能够传达文本的关键信息和主题。

词云图通常由一个平铺的区域组成，其中包含了各种词汇。词汇的大小和颜色可以根据其在文本中的重要性或频率进行调整。通常情况下，出现频率较高的词汇会显示得更大、更醒目，而出现频率较低的

词汇则显示得较小、较不显眼。

词云图在文本分析、舆情监测、主题提取等领域广泛应用。它可以帮助人们快速了解文本的关键内容、主题倾向以及常见的关键词汇。通过可视化呈现词汇，词云图使得文本的分析和理解更加直观和易于理解。

生成词云图的过程通常包括以下步骤：

1. 文本预处理：对文本进行清洗、分词等预处理操作，去除无关词汇和噪声。
2. 词频统计：统计每个词汇在文本中的出现频率。
3. 词云生成：根据词频生成词云图，将常见词汇以较大字体展示，罕见词汇以较小字体展示。
4. 可视化展示：将生成的词云图呈现给用户，以便观察和分析。

分词 jieba 库是一个流行的中文文本处理工具，用于中文分词。它是基于 Python 开发的，提供了高效、可靠的中文分词功能，并且易于使用。

jieba 库具有以下特点和功能：

- 支持中文分词：jieba 库的主要功能是将中文文本进行分词，将连续的中文字符序列切分为一个个有意义的词语。分词是中文自然语言处理的基础步骤，对于文本分析和处理非常重要。
- 多种分词模式：jieba 库支持多种分词模式，包括精确模式、全模式和搜索引擎模式。可以根据不同的需求选择合适的分词模式。
- 支持自定义词典：jieba 库允许用户加载自定义的词典，以增加分词准确性和适应特定领域的词汇。用户可以根据自己的需求添加专业术语、地名、人名等词汇，提高分词效果。
- 并行分词：jieba 库内部使用了多线程来加速分词处理，提高了分词的效率。
- 支持繁体中文分词：jieba 库可以处理繁体中文文本，并提供了针对繁体中文的分词功能。
- 兼容性好：jieba 库兼容 Python 2.x 和 Python 3.x 版本，可以在各种环境中使用。
- 开源社区活跃：jieba 库是开源的，代码托管在 GitHub 上。由于其易用性和高性能，它在中文文本处理领域得到了广泛的应用和支持，拥有活跃的开发者社区。

总之，jieba 库是一个强大且易用的中文分词工具，适用于各种中文文本处理任务，包括文本分析、机器学习、信息检索等。通过使用 jieba 库，可以方便地对中文文本进行分词处理，并进一步进行后续的文本分析和处理。

---

jieba 库提供了两种主要的分词模式：精确模式（默认）和全模式。此外，还提供了一种搜索引擎模式，用于更精细的分词需求。

- 精确模式（精确切分句子，适合文本分析）：该模式下，jieba 会尽可能地将句子切分成精确的词语。它通过基于前缀词典实现最大匹配法来进行分词。

[13]:
```
import jieba

text = "我爱自然语言处理"
seg_list = jieba.cut(text, cut_all=False)
seg_result = " ".join(seg_list)
print(seg_result)
```

```
我 爱 自然语言 处理
```

- 全模式（将句子中所有可能的词语都扫描出来，速度较快）：该模式下，jieba 会将句子中所有可能的词语都进行切分，可能会产生冗余的词语。

[14]:
```
import jieba

text = "我爱自然语言处理"
seg_list = jieba.cut(text, cut_all=True)
seg_result = " ".join(seg_list)
print(seg_result)
```

```
我 爱 自然 自然语言 语言 处理
```

- 搜索引擎模式（在精确模式的基础上，对长词再次切分）：该模式下，jieba 会对较长的词语进行再次切分，提高召回率。

[16]:
```
import jieba

text = "我爱自然语言处理"
seg_list = jieba.cut_for_search(text)
seg_result = " ".join(seg_list)
print(seg_result)
```

```
我 爱 自然 语言 自然语言 处理
```

Recall 和 Accuracy

Recall 在信息检索领域，搜索引擎的召回率（Recall）是衡量搜索引擎检索结果覆盖率的指标。召回率表示在所有相关文档中，搜索引擎成功地检索到了多少个相关文档。

具体而言，召回率是指检索到的相关文档数与所有相关文档数的比例。召回率的计算公式如下：

Recall = 检索到的相关文档数 / 所有相关文档数

召回率的取值范围是 0 到 1 之间，通常以百分比的形式表示。较高的召回率意味着搜索引擎能够更全面地找到相关文档，覆盖了更多的相关信息。

Accuracy 准确率（Accuracy）是指搜索引擎返回的相关文档中真正相关的文档数与返回的总文档数之间的比例。准确率衡量了搜索引擎返回的结果中有多少是正确的。

具体而言，准确率的计算公式如下：

Accuracy = 真正相关的文档数 / 返回的总文档数

准确率的取值范围也是 0 到 1 之间，通常以百分比的形式表示。较高的准确率表示搜索引擎返回的结果中有更多是相关的文档，减少了误报的情况。

准确率和召回率是相互关联的指标。提高准确率可能会导致召回率降低，因为为了减少误报，搜索引擎可能会过滤掉一些相关文档。提高召回率可能会导致准确率降低，因为搜索引擎可能会返回更多的非相关文档。

在搜索引擎评估中，通常需要综合考虑准确率和召回率这两个指标。如果希望结果更准确，可以追求更高的准确率，但可能会降低召回率；如果希望结果更全面，可以追求更高的召回率，但可能会降低准确率。根据具体的应用场景和需求，可以根据准确率和召回率之间的平衡来选择最适合的搜索引擎设置和调优策略。

TF-IDF TF-IDF（Term Frequency-Inverse Document Frequency）是一种常用的文本特征表示方法，用于评估一个词语对于一个文档集合中的某个文档的重要程度或特征权重。

TF（词频）指的是一个词语在**当前文档**中出现的频率。TF 值越高，表示该词语在文档中出现得越频繁。

IDF（逆文档频率）指的是一个词语在**整个文档集合**中的稀有程度。IDF 值越高，表示该词语在整个文档集合中出现得越少，具有更高的独特性。

TF-IDF 的计算公式如下：

TF-IDF = TF * IDF

其中，TF-IDF 值越高，表示该词语在**当前文档**中的重要性越高。这是因为，搜索引擎的核心目标是查找到相关的文档，而对区别文档最有意义的词语应该是那些在文档中出现频率高、而在整个文档集合的其他文档中出现频率少的词语。

TF-IDF 的作用是通过对词语的词频和文档频率进行加权，突出在特定文档中具有较高重要性的词语。常见的应用包括文本分类、信息检索、关键词提取等任务。在文本分类中，TF-IDF 可以帮助识别出在某一类别中具有较高区分度的关键词；在信息检索中，TF-IDF 可以用于计算查询词与文档的匹配程度，从而排序搜索结果。

需要注意的是，TF-IDF 方法并不考虑词语的语义信息，它只关注词频和文档频率。某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF

倾向于过滤掉常见的词语，保留重要的词语。

0.0.2 删除停止词

停止词（Stop words）是在文本处理和信息检索中指那些对文本含义没有重要贡献的常见词语，例如”and”、“the”、“is” 等。这些词语在自然语言中经常出现，但通常不携带特定的语义信息或对文本分类、搜索等任务的结果产生显著影响。

停止词通常是由语言的常用词汇组成的一个固定列表。具体的停止词列表可能因应用场景、语言和任务而异，因为某些词语在特定领域或任务中可能具有重要意义。

在文本处理和信息检索任务中，删除停止词有以下几个主要原因：

1. 减少数据噪音：停止词通常在文本中出现频率很高，但它们对于理解文本的内容和意义并没有太大帮助，只会增加数据的噪音。删除停止词可以减少冗余信息，使得后续文本处理和分析更加集中和有效。
2. 减少存储空间和计算成本：停止词出现频率高，如果保留所有的停止词，会占用较大的存储空间和计算资源。通过删除停止词，可以减少数据的规模和复杂性，提高处理速度和效率。
3. 提高关键信息的权重：在某些文本处理任务中，关键信息的权重对于结果的准确性和可靠性至关重要。删除停止词可以使得关键信息在文本表示中更加突出，提高重要信息的权重，有助于更好地捕捉文本的含义和特征。

获取中文停止词列表一般可以通过以下两种途径：

- 开源停止词库：有一些开源项目提供了中文停止词列表，可以直接下载和使用。其中比较常用的是哈工大停用词表、百度停用词表、中文停用词库等。你可以在这些项目的官方网站或代码库中找到相应的停止词列表文件。
- 自然语言处理库：许多自然语言处理（NLP）库和工具包，如 NLTK、spaCy、jieba 等，提供了预定义的中文停止词列表。你可以使用这些库的内置功能来获取停止词列表。

词云的实例

> 词云（Word Cloud）是一种可视化方式，用于展示一段文本中频繁出现的词汇，以便于观察和分析。词云通过将文本中的词汇按照其出现频率或重要性进行可视化，将常见的词汇以较大的字体大小展示，而罕见的词汇则以较小的字体大小展示。这样，词云图形在一瞥之间能够传达文本的关键信息和主题。

1. 获取文本
2. 简单清洗
3. 调用封装好的工具箱，生成词云

[ ]:
```python
from os import path
from wordCloud import chnSegment
from wordCloud import plotWordcloud

script_folder = globals()['_dh'][0]
print(script_folder)

# 读取文件
with open(path.join(script_folder, 'doc/example.txt')) as f:
    text = f.read()

# 则先进行分词操作
text = chnSegment.word_segment(text)  # 里面调用了 jieba

# 生成词云
plotWordcloud.generate_wordcloud(text)
```

[27]:
```python
# Jupyter Notebook
script_folder = globals()['_dh'][0]
# Python Script
# from os import path
# script_folder = path.dirname(__file__)
print(script_folder)
```

```
/Users/xiejiss/Code/python/python-preparatory-course/day10
```

[20]:
```python
print(path.join(script_folder, 'doc/example.txt'))
```

```
/Users/xiejiss/Code/python/python-preparatory-course/day10/doc/example.txt
```

文件所在目录和工作目录的区别 文件所在目录（File Directory）是指文件在文件系统中的位置路径，表示文件在硬盘上的存储位置。它通常由文件名和文件路径组成，用于唯一标识文件在文件系统中的位置。

工作目录（Working Directory），也称为当前目录（Current Directory），是指在操作系统中当前正在进行操作的进程所在的目录。它是操作系统提供的一个概念，表示进程在执行命令或操作时的参考目录。

两者的区别在于：

文件所在目录是文件在文件系统中的位置，是文件的存储位置信息。而工作目录是进程当前操作的参考目录，是操作系统用来解析相对路径的基准目录。

文件所在目录是静态的，文件位置在创建时确定，并不会随着进程的改变而改变。而工作目录是动态的，随着进程的操作或切换而改变。

文件所在目录是文件的属性，可以被其他进程或用户查询到。工作目录是进程的属性，对其他进程或用户不可见。

在具体的操作中，文件所在目录和工作目录可以是相同的，也可以是不同的。当使用相对路径引用文件时，操作系统会根据当前的工作目录来解析路径，找到对应的文件。如果工作目录与文件所在目录不同，需要使用相对路径或绝对路径来准确定位文件。

[21]:
```python
import os

# 获取当前工作目录
working_directory = os.getcwd()

print("当前工作目录: ", working_directory)
```

```
当前工作目录:  /Users/xiejiss/Code/python/python-preparatory-course/day10
```

[25]:
```python
try:
    os.mkdir("script")
except:
    pass

with open("./script/test.py", "w") as f:
    f.write("""
import os

# 获取当前工作目录
working_directory = os.getcwd()

print("当前工作目录: ", working_directory)

# Jupyter Notebook
# script_folder = globals()['_dh'][0]
# Python Script
from os import path
script_folder = path.dirname(__file__)
print("文件所在目录: ", script_folder)
""")
```

[26]:
```python
!python3 ./script/test.py
```

```
当前工作目录:  /Users/xiejiss/Code/python/python-preparatory-course/day10
文件所在目录:  /Users/xiejiss/Code/python/python-preparatory-course/day10/./script
```

path.join　path.join() 是 Python 中 os.path 模块中的一个函数，用于将多个路径片段拼接成一个完整的路径。

path.join() 函数会根据操作系统的规则自动处理路径分隔符，确保生成的路径在不同操作系统上都是正确的。这使得代码在不同平台上的兼容性更好。

[29]:
```python
from os import path

path1 = "."
path2 = "file.txt"

full_path = path.join(path1, path2)

print("拼接后的完整路径: ", full_path)
```

```
拼接后的完整路径:  ./file.txt
```

path.abspath　path.abspath() 是 Python 中 os.path 模块中的一个函数，用于获取给定路径的绝对路径。

path.abspath() 函数会将相对路径转换为绝对路径。如果给定的路径已经是绝对路径，则返回原始路径。

需要注意的是，path.abspath() 函数只是返回给定路径的绝对路径字符串，并不会检查路径的存在性或有效性。它只是对路径字符串进行处理，确保返回的路径为绝对路径形式。

[30]:
```python
from os import path

path1 = path.abspath(".")
path2 = "file.txt"

full_path = path.join(path1, path2)

print("拼接后的完整路径: ", full_path)
```

```
拼接后的完整路径:  /Users/xiejiss/Code/python/python-preparatory-course/day10/file.txt
```

0.0.3 情感极性分析

情感极性分析（Sentiment Polarity Analysis）是一种文本分析技术，用于确定给定文本表达的情感倾向或情感极性。它旨在自动判断文本的情感是正面、负面还是中性。

情感极性分析的一般流程如下：

1. 数据准备：收集或获取待分析的文本数据，可以是用户评论、社交媒体帖子、新闻文章等。
2. 文本预处理：对文本数据进行预处理，包括分词、去除停止词、词干化或词形还原等步骤，以便更好地表示文本的含义和特征。
3. 特征提取：从预处理后的文本中提取有意义的特征，常用的特征表示方法包括词袋模型（Bag of Words）、词频-逆文档频率（TF-IDF）等。
4. 构建情感分类模型：使用标注好的情感数据集训练机器学习或深度学习模型，例如朴素贝叶斯、支持向量机、逻辑回归、循环神经网络（RNN）、卷积神经网络（CNN）等。
5. 情感分类：将待分析的文本输入训练好的模型，进行情感分类预测。通常输出的情感极性标签为正面、负面或中性。
6. 结果解释和评估：对分类结果进行解释和评估，可以使用评估指标如准确率、精确度、召回率、F1值等来衡量模型的性能。

但我们作为初学者，在这里可以直接调用已经编写好的工具箱，可以简单看看情感极性分析的效果。

[42]:
```
!pip install loguru
```

```
Collecting loguru
  Downloading loguru-0.7.0-py3-none-any.whl (59 kB)

60.0/60.0 kB 187.0 kB/s eta 0:00:00 kB/s eta
0:00:01:01
Installing collected packages: loguru
Successfully installed loguru-0.7.0

[notice] A new release of pip
available: 22.3.1 -> 23.2.1
[notice] To update, run:
pip install --upgrade pip
```

0.0.4 规则的解决思路

1. 中文情感极性分析，文本切分为段落，再切词，通过情感词标识出各个词语的情感极性，包括积极、中立、消极。
2. 结合句子结构（包括连词、否定词、副词、标点等）给各情感词语的情感极性赋予权重，然后加权求和得到文本的情感极性得分。

3. 优点：泛化性好，规则可扩展性强，所有领域通用。
4. 缺点：规则词典收集困难，专家系统的权重设定有局限，单一领域准确率相比模型方法低。

0.0.5 模型的解决思路

1. 常见的NLP 文本分类模型均可，包括经典文本分类模型（LR、SVM、Xgboost 等）和深度文本分类模型（TextCNN、Bi-LSTM、BERT 等）。
2. 优点：单一领域准召率高。
3. 缺点：不通用，有标注数据的样本收集困难，扩展性弱。

[2]:
```
import pysenti

texts = ["苹果是一家伟大的公司",
         "土豆丝很好吃",
         "土豆丝很难吃"]
for i in texts:
    r = pysenti.classify(i)
    print(i, r['score'], r)
    print()
```

```
苹果是一家伟大的公司 3.4346924811096997 {'score': 3.4346924811096997, 'sub_clause0':
{'score': 3.4346924811096997, 'sentiment': [{'key': '苹果', 'adverb': [],
'denial': [], 'value': 1.37846341627, 'score': 1.37846341627}, {'key': '是',
'adverb': [], 'denial': [], 'value': -0.252600480826, 'score': -0.252600480826},
{'key': '一家', 'adverb': [], 'denial': [], 'value': 1.48470161748, 'score':
1.48470161748}, {'key': '伟大', 'adverb': [], 'denial': [], 'value':
1.14925252286, 'score': 1.14925252286}, {'key': '的', 'adverb': [], 'denial': [],
'value': 0.0353323193687, 'score': 0.0353323193687}, {'key': '公司', 'adverb': [],
'denial': [], 'value': -0.360456914043, 'score': -0.360456914043}],
'conjunction': []}}

土豆丝很好吃 2.294311221077 {'score': 2.294311221077, 'sub_clause0': {'score':
2.294311221077, 'sentiment': [{'key': '土豆丝', 'adverb': [], 'denial': [],
'value': 0.294892711165, 'score': 0.294892711165}, {'key': '很', 'adverb': [],
'denial': [], 'value': 0.530242664632, 'score': 0.530242664632}, {'key': '好吃',
'adverb': [], 'denial': [], 'value': 1.46917584528, 'score': 1.46917584528}],
'conjunction': []}}

土豆丝很难吃 -2.381874203563 {'score': -2.381874203563, 'sub_clause0': {'score':
-2.381874203563, 'sentiment': [{'key': '土豆丝', 'adverb': [], 'denial': [],
```

```
'value': 0.294892711165, 'score': 0.294892711165}, {'key': '很', 'adverb': [],
'denial': [], 'value': 0.530242664632, 'score': 0.530242664632}, {'key': '难吃',
'adverb': [], 'denial': [], 'value': -3.20700957936, 'score': -3.20700957936}],
'conjunction': []}}
```

[7]:
```
from pysenti import ModelClassifier

texts = ["苹果是一家伟大的公司",
         "土豆丝很好吃",
         "垃圾，在酒店中应该是很差的！",
         "我们刚走过一个烧烤店",
         "土豆丝很难吃"]

m = ModelClassifier()
for i in texts:
    r = m.classify(i)
    print(i, r)
    print()
```

```
苹果是一家伟大的公司 {'positive_prob': 0.6819638428483796, 'negative_prob':
0.31803615715162037}

土豆丝很好吃 {'positive_prob': 0.6008313281872775, 'negative_prob':
0.39916867181272253}

垃圾，在酒店中应该是很差的！ {'positive_prob': 0.06056508633079882, 'negative_prob':
0.9394349136692012}

我们刚走过一个烧烤店 {'positive_prob': 0.8248988123579972, 'negative_prob':
0.17510118764200278}

土豆丝很难吃 {'positive_prob': 0.2831455391498756, 'negative_prob':
0.7168544608501244}
```

[ ]:
```
# 使用情感极性分析，分析电影的一句话评语
```

0.0.6 配色

[6]:
```
import colorgram
```

[10]:
```
colors = colorgram.extract('example.jpg', 6)
print(colors)

first_color = colors[0]
rgb = first_color.rgb
hsl = first_color.hsl
proportion = first_color.proportion

r = first_color.rgb.r
g = first_color.rgb.g
b = first_color.rgb.b
print(rgb, hsl, proportion, r, g, b)

# 生成 HTML 展示图片和所有颜色
```

```
[<colorgram.py Color: Rgb(r=219, g=230, b=244), 50.51671056314275%>,
<colorgram.py Color: Rgb(r=119, g=174, b=210), 24.063880735547055%>,
<colorgram.py Color: Rgb(r=34, g=108, b=159), 7.934377758480871%>, <colorgram.py
Color: Rgb(r=175, g=186, b=220), 5.96047164191958%>, <colorgram.py Color:
Rgb(r=213, g=243, b=237), 5.880812096768406%>, <colorgram.py Color: Rgb(r=249,
g=248, b=244), 5.64374720414134%>]
Rgb(r=219, g=230, b=244) Hsl(h=151, s=135, l=231) 0.5051671056314275 219 230 244
```

[ ]:
```

```