基于 SVM 的中文评论情感分类研究

李奕则

2025 年 6 月 27 日

摘要

本文利用支持向量机（SVM）模型对中文网络评论进行情感倾向性识别。通过 jieba 分词与 TF-IDF 特征提取构建高维稀疏向量空间，并训练线性核 SVM 分类器以判断评论倾向。实验结果表明，在自建数据集（online_shopping.csv）上准确率达到 90.68%，表明该方法在中文情感分类任务中具有良好的适用性。同时，我们设计了基于模型的图形化预测界面，提升其可交互性与实用性。

关键词：支持向量机；情感分类；中文评论；TF-IDF；图形界面；文本挖掘

目录

前言

本文档有线上版本，存放于 Overleaf 网站，项目网址如下：

https://cn.overleaf.com/read/nbcfnnmhwhbn#36f493

由于各种超链接以及代码源文件没法在纸质版上完整呈现，建议浏览原先项目文档。

数据集 `online_shopping_10_cats.csv` 来源于 GitHub，下载地址：
https://github.com/SophonPlus/ChineseNlpCorpus/raw/master/datasets/online_shopping_10_cats

本文档的全部代码放在 GitHub 平台，地址如下：https://github.com/liyizenb/SVM-test

一 引言

在信息爆炸的时代，用户通过网络平台发布了海量评论文本，这些文本反映了其对商品、服务、社会事件乃至公共政策的真实态度与主观情感。这些用户生成内容具有极高的价值，对其进行有效的情感分析不仅有助于商家深入了解消费者的需求，改进产品与服务，也为政府进行舆情监控、社会治理以

及大众传播研究提供了技术支撑。在大数据背景下，如何从繁杂的评论文本中自动提取情感倾向，已成为自然语言处理领域的重要研究方向之一。

情感分析任务中，基于机器学习的分类模型因其建模能力强、可推广性好而被广泛应用。支持向量机（SVM）作为一种典型的小样本监督学习方法，尤其适用于高维稀疏的文本分类场景，近年来在情感分类、观点识别等任务中展现出良好性能。相比于深度学习对大规模数据与计算资源的依赖，SVM 具有结构简洁、鲁棒性强等优势，特别适合中小型数据集的建模分析。

本研究基于网络评论数据，融合 TF-IDF 文本向量表示与人工构建的情感词典特征（包括积极词数量、消极词数量与情感评分），构建了一个多特征融合的 SVM 文本情感分类模型。通过对评论文本进行分词、去除停用词、提取双字/三字词组等处理，并结合人工标注的积极与消极情感词汇，提取结构化特征后，训练支持向量机模型以识别评论的情感倾向。实验中还引入训练进度显示与训练时间统计功能，提升了训练过程的可控性与可解释性。最终模型在测试集上取得了较高的准确率，验证了 TF-IDF 与情感特征融合的有效性，也表明 SVM 模型在中文文本情感分析任务中仍具有现实可行性与应用潜力。

该方法为中小规模舆情数据的自动化分析提供了一种高效、低成本的技术路径，可拓展应用于电商平台、政务服务反馈分析、微博舆情监测等多个实际场景。

二 方法与建模

二.1 文本预处理与特征表示

为提升情感分类效果，本文对原始评论文本进行了系统的预处理与多维特征提取。首先，利用 jieba 工具对中文文本进行分词，并结合自定义停用词表，过滤掉无实际语义贡献的功能性词汇。预处理后的文本经由 TF-IDF 方法转换为高维稀疏向量表示。设词汇表维度为 d，每条评论文本被表示为向量 $\mathbf{x}_{\text{tfidf}} \in \mathbb{R}^d$，其中第 i 维的 TF-IDF 权重定义如下：

$$\text{TF-IDF}(w_i, x) = \text{TF}(w_i, x) \cdot \log\left(\frac{N}{\text{DF}(w_i)}\right) \tag{1}$$

其中，$\text{TF}(w_i, x)$ 表示词语 w_i 在文本 x 中出现的频率，$\text{DF}(w_i)$ 为包含词语 w_i 的文档数，N 为语料库中文档总数。

此外，本文引入人工构建的情感词典，分别统计评论中出现的积极词数量、消极词数量以及两者的差值（即情感得分），作为补充特征向量。设积极词集合为 $\mathcal{P}$，消极词集合为 $\mathcal{N}$，则每条评论文本还被表示为一个情感特征向量：

$$\mathbf{x}_{\text{emo}} = [\text{pos\_count}, \text{neg\_count}, \text{emotion\_score}]^\top$$

其中，$\text{pos\_count} = \sum_{w \in x} \mathbb{I}(w \in \mathcal{P})$，$\text{neg\_count} = \sum_{w \in x} \mathbb{I}(w \in \mathcal{N})$，而 $\text{emotion\_score} = \text{pos\_count} - \text{neg\_count}$。

最终，文本的表示为 TF-IDF 特征与情感特征的拼接向量，即

$$\mathbf{x} = [\mathbf{x}_{\text{tfidf}};\ \mathbf{x}_{\text{emo}}] \in \mathbb{R}^{d+3}$$

该多特征组合策略兼顾了语义信息与情感信号，为后续的 SVM 情感分类模型训练提供了更为丰富的表达基础。

二.2　支持向量机原理

支持向量机（Support Vector Machine, SVM）是一种经典的二分类模型，广泛应用于文本分类、舆情识别与情感分析等任务中，特别适用于高维、稀疏、样本量中等的场景。其核心思想是通过构造一个最优超平面，将样本集尽可能正确且最大间隔地分为两个类别。

设有训练样本集 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，其中 $\mathbf{x}_i \in \mathbb{R}^d$ 表示第 i 条评论的特征向量，$y_i \in \{1, -1\}$ 为其情感标签。SVM 试图构造一个线性决策函数：

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

使得正类样本满足 $f(\mathbf{x}_i) \geq +1$，负类样本满足 $f(\mathbf{x}_i) \leq -1$，从而使几何间隔最大。为适应现实中存在部分噪声与误标记的情况，引入松弛变量 $\xi_i \geq 0$，并以惩罚参数 C 控制对误分类样本的容忍程度，从而得到软间隔 SVM 的优化目标函数：

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i \tag{2}$$

$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n$$

为便于求解，通常将上述原始问题（Primal Problem）转化为其对偶形式。通过引入拉格朗日乘子 $\alpha_i \geq 0$，对原问题构造拉格朗日函数：

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right] - \sum_{i=1}^n \mu_i\xi_i$$

对 $\mathbf{w}$、b 与 ξ_i 求偏导并令其为零，代入原函数可得其对偶问题（Dual Problem）形式：

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle \tag{3}$$

$$\text{subject to } \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$

其中，$\langle \mathbf{x}_i, \mathbf{x}_j\rangle$ 表示样本特征的内积，当使用核函数 $K(\mathbf{x}_i, \mathbf{x}_j)$ 代替时，即可推广到非线性情形。

该对偶问题为凸二次规划问题（Convex Quadratic Programming），可以通过现代数值优化算法在计算机中高效求解。常用方法包括序列最小优化（Sequential Minimal Optimization, SMO）、拟牛顿法、坐标下降法等。本文所采用的 `scikit-learn` 库中的 `SVC` 类默认使用基于 LibSVM 实现的 SMO 算法，该算法通过每次选择两个变量进行局部最优化，逐步逼近最优解，在中小型数据集上具有良好的收敛性与计算效率。

最终，解得拉格朗日乘子 α_i 后，可计算模型参数：

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \quad b = y_k - \sum_{i=1}^n \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_k\rangle$$

其中 k 为任一满足 $0 < \alpha_k < C$ 的支持向量下标。对新样本 $\mathbf{x}$，其预测结果由如下公式决定：

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right)$$

该形式表明，最终决策函数仅依赖于训练集中少数几个支持向量的线性组合，具有良好的泛化能力与稀疏表示特性。

二.3 参数选择

考虑到原始数据集包含 62,275 条评论，直接在全量数据上进行 SVM 的训练和超参数调优计算量巨大，训练时间难以接受。因此，本文首先采用随机抽样方法，从原始数据中随机抽取 5,000 条样本进行后续的模型训练与参数选择。该采样过程确保数据的多样性与代表性，同时极大降低了计算资源消耗，为超参数搜索提供了合理的计算环境。

在抽样数据集上，本文采用网格搜索（GridSearchCV）结合三折交叉验证，对以下关键参数进行调节：

- 正则化参数 $C \in \{0.5, 1, 5\}$，用于平衡训练误差和模型复杂度；
- 核函数类型：线性核（linear）与径向基函数核（RBF）两类，兼顾线性与非线性数据结构；
- 核函数参数 $\gamma \in \{\text{scale}, \text{auto}\}$，控制 RBF 核的影响范围。

该方法通过多轮交叉验证，评估每组参数在验证集上的表现，自动选择最优参数组合以保证模型的泛化性能。抽样与调参相结合的策略，不仅有效减少了训练时间，也避免了因全量数据训练导致的过拟合风险，为后续在完整数据上的模型应用和误差分析奠定了坚实基础。

三 实验设计与结果

三.1 数据集

本实验使用公开数据集 `online_shopping_10_cats.csv`，该数据集来源于 GitHub，下载地址：https://github.com/SophonPlus/ChineseNlpCorpus/raw/master/datasets/online_shopping_10_cats

这个数据集包含来自十个商品类别（书籍、平板、手机、水果、洗发水、热水器、蒙牛、衣服、计算机、酒店）的约 6 万条用户评论及其对应的情感标签（正向和负向评论数量大致相当）。

数据经过整合与去重处理，覆盖多个电商平台，适用于情感倾向性分析任务。各类别样本分布情况见表 1。

表 1: 数据集中各类别评论的样本分布情况

类别	总体样本数	正例样本数	负例样本数
书籍	3851	2100	1751
平板	10000	5000	5000
手机	2323	1165	1158
水果	10000	5000	5000
洗发水	10000	5000	5000
热水器	575	475	100
蒙牛	2033	992	1041
衣服	10000	5000	5000
计算机	3992	1996	1996
酒店	10000	5000	5000

三.2 实验基础设置

本实验数据预处理方法与参数选择原因已在前文二.1与二.3中展示，不再赘述。关键参数配置如表 2 所示。

表 2: 实验关键参数设置

参数项	值	说明
特征类型	TF-IDF + 情感词典特征	拼接组合
TF-IDF 最大维度	8000	1-2 元语法，min_df=2, max_df=0.9
情感词典特征	3 维	正词数、负词数、情感分数
训练集比例	80%	stratify 分层抽样
SVM 核函数	linear	线性可分假设
惩罚参数 C	1	误分类惩罚系数
核函数参数 γ	scale	sklearn 默认自适应设置
模型保存方式	joblib	保存模型与向量器
进度可视化	tqdm	显示文本处理进度条

三.3 消融实验与对比实验

为验证模型在文本情感分类中的有效性与泛化能力，本文分别从整体建模与分组训练两个角度开展对比实验，同时引入情绪词典方法作为基准，具体设置如下：

整体模型对比分析

首先，在全体样本基础上，分别使用支持向量机（SVM）与人工构建的情绪词典方法进行情感分类建模。其中，SVM 模型结合 TF-IDF 向量（8000 维，双词组支持）与人工定义的情绪特征（正向词数、负向词数、情绪差值），采用线性核进行训练。实验结果如表 3 所示。

表 3: 整体模型性能对比

方法	准确率	用时（秒）	备注
SVM（TF-IDF）	90.60%	446.57	除预处理外 SVM 参数全默认
SVM（TF-IDF + 情绪特征）	90.75%	447.38	含 8000 维 TF-IDF + 3 维情绪词统计
情绪词典打分法	84.89%	~0.5	使用情绪词、否定词、程度副词手动评分

由上表可见，基于机器学习的 SVM 方法在准确率上明显优于传统情绪词典打分法，且召回率、F1 值等指标亦表现良好（宏平均召回率为 90.76%，宏平均 F1 值为 90.75%，加权平均 F1 值为 90.75%），表明其对复杂语义具有更强的表达与判别能力。

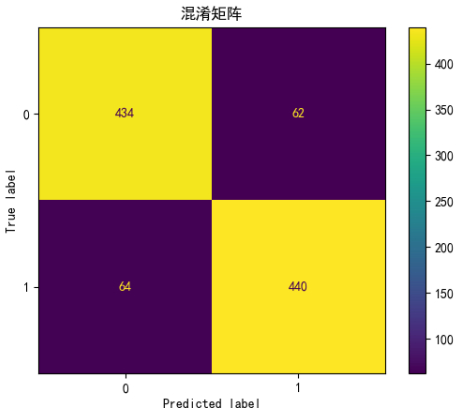

图 1: 部分样本混淆矩阵图

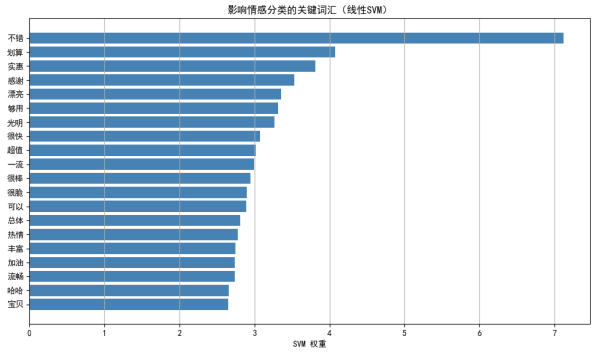

图 2: 关键词汇影响力图

我们还绘制了 SVM 的混淆矩阵图（图 1）和影响情感分类的关键词汇图（图 2），可以看出“不错”一词对评论的正负项分类有很大的影响，以断层优势领先。

每类商品独立训练实验

考虑到不同类别商品评论存在语言风格差异，为进一步评估模型对不同类别数据的适应能力，我们对每个商品类别分别独立训练一个 SVM 模型，并统计其准确率与训练耗时，结果如表 4 所示。

表 4: 按商品类别独立训练 SVM 的分类性能

类别	样本数	准确率	训练耗时（秒）
书籍	3,851	89.88%	0.90
平板	10,000	92.05%	1.71
手机	2,323	92.69%	0.26
水果	10,000	90.50%	1.96
洗发水	10,000	92.30%	1.63
热水器	575	88.70%	0.02
蒙牛	2,033	92.38%	0.10
衣服	10,000	94.65%	1.28
计算机	3,992	91.99%	0.58
酒店	10,000	90.30%	3.92

可见，大多数类别在独立训练条件下表现良好，尤其“衣服”与“手机”等类别的准确率超过 92%，说明该方法在语义集中、词汇分布稳定的场景下具备优秀的情感判别能力。相比之下，“热水器”类别样本较少，模型表现略弱，提示样本量对模型稳定性仍有影响。

三.3.1 小结

综合对比可得，SVM 模型不仅整体表现优于情绪词典方法，且在多数类别中展现出稳定、较高的识别准确率。此外，分类别建模在一定程度上能够提升局部分类性能，尤其在类别分布异质性较强的场景下效果更为明显。因此，结合先验信息进行分组建模是值得探索的方向之一。

三.4 实时分析（一个好玩的内容）

为提升本模型的可视化应用效果，我设计了一个基于 Tkinter 的交互式窗口，用户可输入任意文本并获得实时情感判定结果。在之前训练各种模型的时候我均有意将模型文件保存，因此可以很简单的通过对接模型实现这个功能，界面如图所示。

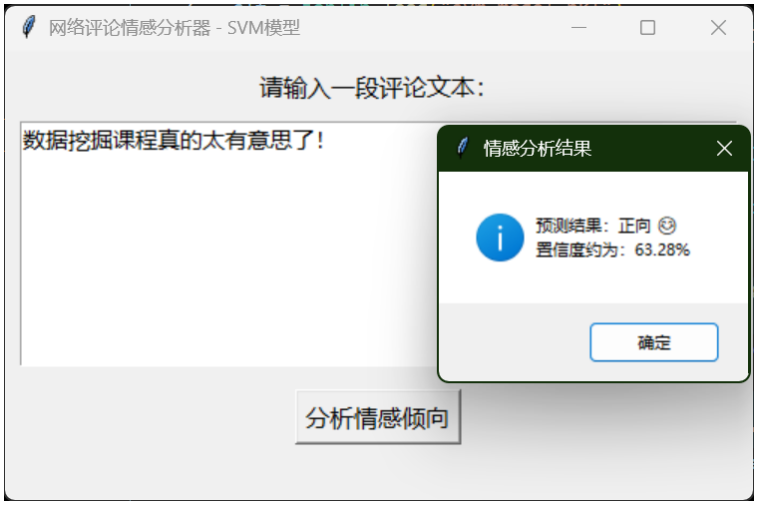

图 3: 实时情感分析界面

四 SVM 方法优势与不足分析

四.1 优势

- **适合高维稀疏数据：** 文本数据特征维度高、非线性强，SVM 在线性核下依然具有良好表现；
- **较强的泛化能力：** 小样本下表现稳定，避免过拟合；
- **理论基础扎实：** 源于结构风险最小化原理，具备几何解释。

四.2 不足

- **训练时间较长：** 尤其在数据量大、维度高时，训练耗时显著，例如无法直接对全样本进行超参数；
- **对参数敏感：** C、核函数参数需精心调节；

五 结论与学习所得

本文基于 SVM 方法实现了对中文评论的情感分类任务，准确率超过 90%，并通过 TF-IDF 与权重可视化解释模型决策机制。我们进一步分析了不同参数设置下的模型表现，并讨论了 SVM 的优势与局限。此外，通过构建 Tkinter 图形界面，提升了模型的交互性与现实应用价值。

- 了解了 SVM 适合高维稀疏数据，并且具有较强的泛化能力；
- 评论信息的情绪表达具有较强的线性性，从 SVM 最优参数为线性核、情绪词典这种纯线性方法效果也达到 80+% 可以看出；
- 令人出乎意料的，对情绪影响最大的词汇是“不错”，推测虽然其语气不够强烈，但是由于“不错”很少发生在反讽的例子中，从而达到了与负样本的明显分离；
- 评论的识别难度与商品的实用性和样本数量挂钩。

六 代码

表 5: 各程序功能概览

文件名	功能说明
`dic.py`	基于情绪词典的简单规则方法进行全样本情感分类预测
`SVM.py`	基于 TF-IDF 特征的 SVM 模型，在全样本上训练和预测
`grid_search.py`	在 5000 条样本上进行超参数搜索（如 C、kernel、gamma）
`final_SVM.py`	使用调优后的参数，结合 TF-IDF 与情绪词典特征进行全样本预测
`each_cat.py`	针对每个商品类别分别训练 SVM 模型，输出各类别准确率

六.1 具体代码展示

code/dic.py

```python
import pandas as pd
import jieba
from sklearn.metrics import accuracy_score

# 1. 加载数据
df = pd.read_csv("online_shoping.csv")
df['review'] = df['review'].fillna('').astype(str)

# 2. 构建词典（示例，建议用更全词典替换）
positive_words = {'喜欢', '满意', '棒', '好', '优秀', '值得', '美丽', '快乐', '自由', '真理', '进步'}
```

```
negative_words = {'差', '糟糕', '坏', '失望', '低级', '过于', '一败涂地'}
negation_words = {'不', '没', '无', '未', '否', '别'}
degree_words = {
    '非常': 2.0, '特别': 2.0, '十分': 2.0,
    '很': 1.5, '较': 1.2, '有点': 0.8, '稍微': 0.5
}

# 3. 情感分析函数
def analyze_sentiment(text):

    if pd.isnull(text):  # 如果为空
        return 2  # 视为中性
    text = str(text)  # 确保是字符串

    words = list(jieba.cut(text))
    score = 0
    i = 0
    while i < len(words):
        word = words[i]
        base = 0
        if word in positive_words:
            base = 1
        elif word in negative_words:
            base = -1

        if base != 0:
            neg = 0
            degree = 1
            for j in range(max(0, i - 3), i):
                if words[j] in negation_words:
                    neg += 1
                elif words[j] in degree_words:
                    degree *= degree_words[words[j]]
            if neg % 2 == 1:
                base *= -1
            base *= degree
        score += base
        i += 1

```

```
    if score > 0.5:
        return 1
    elif score < -0.5:
        return 0
    else:
        return 2  # 中性（暂不参与准确率计算）

# 4. 应用分析并比对真实标签
df['pred'] = df['review'].apply(analyze_sentiment)

# 只保留二分类样本用于验证准确率
filtered_df = df[df['pred'] != 2]
y_true = filtered_df['label']
y_pred = filtered_df['pred']

# 5. 输出准确率
acc = accuracy_score(y_true, y_pred)
print(f"情感分类准确率: {acc:.2%}")
print(filtered_df[['cat', 'label', 'pred', 'review']].head())
```

code/SVM.py

```
import pandas as pd
import jieba
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
import time
import joblib
from tqdm import tqdm

# 读取数据集
data = pd.read_csv("online_shoping.csv", encoding="utf-8-sig")

# 分词处理
def tokenize(text):
    return " ".join(jieba.cut(str(text)))

print("正在分词...")
```

```
data['cut_review'] = list(tqdm(map(tokenize, data['review']), total=len(data)))

# TF-IDF 特征提取
print("提取TF-IDF特征...")
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(data['cut_review'])
y = data['label']

# 数据集划分
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 模型训练 + 时间统计
print("开始训练SVM模型...")
start_time = time.time()

clf = svm.SVC(kernel='linear', verbose=True)
clf.fit(X_train, y_train)

end_time = time.time()
print(f"\n训练完成, 用时: {end_time - start_time:.2f} 秒")

# 保存模型和TF-IDF向量器
joblib.dump(clf, "svm_model.pkl")
joblib.dump(vectorizer, "tfidf_vectorizer.pkl")
print("模型和向量器已保存为 'svm_model.pkl' 和 'tfidf_vectorizer.pkl'")

# 预测与评估
y_pred = clf.predict(X_test)
print(f"准确率: {accuracy_score(y_test, y_pred):.4f}")
print(classification_report(y_test, y_pred, digits=4))

# 特征词可视化 (如果是线性核)
if hasattr(clf, "coef_"):
    coefs = clf.coef_.toarray().flatten()
    top_n = 20
    top_indices = coefs.argsort()[-top_n:]
    words = [vectorizer.get_feature_names_out()[i] for i in top_indices]
```

```
    weights = coefs[top_indices]

    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False

    plt.figure(figsize=(10, 6))
    plt.barh(words, weights, color="steelblue")
    plt.xlabel("SVM 权重")
    plt.title("影响情感分类的关键词汇（线性SVM）")
    plt.grid(True, axis='x')
    plt.tight_layout()
    plt.savefig("important_words.png")
    plt.show()
    print("重要特征词图像已保存为 'important_words.png'")
```

code/grid_search.py

```
import pandas as pd
import jieba
import re
import time
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay
from scipy.sparse import hstack
import matplotlib.pyplot as plt
import joblib
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# ----------------------------------
# 0. 随机抽样函数
# ----------------------------------
def sample_csv(input_file, output_file, n, random_state=42):
    df = pd.read_csv(input_file, encoding='utf-8-sig')
    sampled_df = df.sample(n=n, random_state=random_state)
```

```
    sampled_df.to_csv(output_file, index=False, encoding='utf-8-sig')
    print(f"  已从 {input_file} 中随机抽取 {n} 行，保存为 {output_file}")
    return sampled_df

# ---------------------------------
# 1. 停用词 & 情感词典加载
# ---------------------------------
with open("stopwords.txt", encoding="utf-8") as f:
    stopwords = set([line.strip() for line in f])

positive_words = {'喜欢', '满意', '棒', '好', '优秀', '值得', '快乐', '真理', '进步'}
negative_words = {'差', '糟糕', '坏', '失望', '低级', '失败', '一败涂地'}

# ---------------------------------
# 2. 文本清洗与情感特征
# ---------------------------------
def clean_text(text):
    text = re.sub(r"[^\u4e00-\u9fa5]", " ", text)
    words = jieba.cut(text)
    words = [w for w in words if w.strip() and w not in stopwords]
    return " ".join(words)

def emotion_features(text):
    tokens = text.split()
    pos_count = sum(1 for t in tokens if t in positive_words)
    neg_count = sum(1 for t in tokens if t in negative_words)
    return pd.Series([pos_count, neg_count, pos_count - neg_count])

# ---------------------------------
# 主程序
# ---------------------------------
def main():
    input_file = "online_shoping.csv"
    sampled_file = "sampled_data.csv"
    sample_size = 5000

    # Step 0: 抽样数据
    if not os.path.exists(sampled_file):
```

```
        df = sample_csv(input_file, sampled_file, n=sample_size)
    else:
        df = pd.read_csv(sampled_file, encoding='utf-8-sig')
        print(f"  已读取现有文件: {sampled_file}")

    df['review'] = df['review'].fillna('').astype(str)

    print("  正在清洗文本...")
    tqdm.pandas()
    df["cut_review"] = df["review"].progress_apply(clean_text)

    print("  提取情感词典特征...")
    df[["pos_count", "neg_count", "emotion_score"]] = df["cut_review"].
    apply(emotion_features)

    # Step 3: 特征提取
    print("  提取 TF-IDF 特征...")
    tfidf = TfidfVectorizer(
        max_features=8000,
        min_df=2,
        max_df=0.9,
        ngram_range=(1, 2)
    )
    X_tfidf = tfidf.fit_transform(df['cut_review'])
    X_dict = df[['pos_count', 'neg_count', 'emotion_score']]
    X = hstack([X_tfidf, X_dict.values])
    y = df['label']

    # Step 4: 划分数据集
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y)

    # Step 5: 模型训练与网格搜索
    print("  训练模型中...")
    start_time = time.time()
    param_grid = {
        'C': [0.5, 1, 5],
        'kernel': ['linear', 'rbf'],
        'gamma': ['scale', 'auto']
```

```
    }
    grid = GridSearchCV(SVC(), param_grid, cv=3, n_jobs=-1, verbose=1)
    grid.fit(X_train, y_train)
    end_time = time.time()

    clf = grid.best_estimator_
    print(f"  训练完成! 耗时 {end_time - start_time:.2f} 秒")
    print("  最优参数: ", grid.best_params_)

    # Step 6: 模型评估
    y_pred = clf.predict(X_test)
    print(f"\n  准确率: {accuracy_score(y_test, y_pred):.4f}")
    print(classification_report(y_test, y_pred, digits=4))

    cm = confusion_matrix(y_test, y_pred)
    ConfusionMatrixDisplay(cm).plot()
    plt.title("混淆矩阵")
    plt.show()

    # Step 7: 模型保存
    joblib.dump(clf, "svm_model_with_lexicon.pkl")
    joblib.dump(tfidf, "tfidf_vectorizer_with_lexicon.pkl")
    print("  模型和向量器已保存为 'svm_model_with_lexicon.pkl' 和 '
    tfidf_vectorizer_with_lexicon.pkl'")

if __name__ == "__main__":
    main()
```

code/final_SVM.py

```
import pandas as pd
import jieba
import re
import time
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix, ConfusionMatrixDisplay
from scipy.sparse import hstack
```

```
import matplotlib.pyplot as plt
import joblib

# ---------------------------------
# 1. 停用词 & 情感词典加载
# ---------------------------------
with open("stopwords.txt", encoding="utf-8") as f:
    stopwords = set([line.strip() for line in f])

positive_words = {'喜欢', '满意', '棒', '好', '优秀', '值得', '快乐', '真理', '进步'}
negative_words = {'差', '糟糕', '坏', '失望', '低级', '失败', '一败涂地'}

# ---------------------------------
# 2. 数据加载与处理
# ---------------------------------
df = pd.read_csv("online_shoping.csv", encoding="utf-8-sig")
df['review'] = df['review'].fillna('').astype(str)

def clean_text(text):
    text = re.sub(r"[^\u4e00-\u9fa5]", " ", text)  # 仅保留中文
    words = jieba.cut(text)
    return " ".join(w for w in words if w not in stopwords and w.strip())

def emotion_features(text):
    tokens = text.split()
    pos_count = sum(1 for t in tokens if t in positive_words)
    neg_count = sum(1 for t in tokens if t in negative_words)
    return pd.Series([pos_count, neg_count, pos_count - neg_count])

print("正在清洗文本并提取特征...")
tqdm.pandas()
df["cut_review"] = df["review"].progress_apply(clean_text)
df[["pos_count", "neg_count", "emotion_score"]] = df["cut_review"].apply(emotion_features)

# ---------------------------------
# 3. TF-IDF + 情感特征合并
# ---------------------------------
```

```
vectorizer = TfidfVectorizer(max_features=8000, ngram_range=(1, 2), min_df=2, max_df=0.9)
X_tfidf = vectorizer.fit_transform(df["cut_review"])
X_dict = df[["pos_count", "neg_count", "emotion_score"]]
X = hstack([X_tfidf, X_dict.values])
y = df["label"]

# ----------------------------------
# 4. 划分数据集
# ----------------------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# ----------------------------------
# 5. 使用默认线性核 SVM 训练
# ----------------------------------
print("开始训练 SVM...")
start_time = time.time()

clf = SVC(kernel="linear", C=1, gamma='scale')
clf.fit(X_train, y_train)

end_time = time.time()
print(f"训练完成, 用时 {end_time - start_time:.2f} 秒")

# ----------------------------------
# 6. 模型评估
# ----------------------------------
y_pred = clf.predict(X_test)
print(f"\n准确率: {accuracy_score(y_test, y_pred):.4f}")
print(classification_report(y_test, y_pred, digits=4))

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(cm).plot()
plt.title("混淆矩阵")
plt.show()

# ----------------------------------
# 7. 模型保存
```

```
# ----------------------------------
joblib.dump(clf, "svm_model_default.pkl")
joblib.dump(vectorizer, "tfidf_vectorizer_default.pkl")
print("模型和向量器已保存为 'svm_model_default.pkl' 和 'tfidf_vectorizer_default.pkl'")
```

code/each_cat.py

```
import pandas as pd
import jieba
import re
import time
from tqdm import tqdm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from scipy.sparse import hstack
import joblib

# ---------------------------------- 停用词 & 情感词典加载 ----------------------------------
with open("stopwords.txt", encoding="utf-8") as f:
    stopwords = set([line.strip() for line in f])

positive_words = {'喜欢', '满意', '棒', '好', '优秀', '值得', '快乐', '真理', '进步'}
negative_words = {'差', '糟糕', '坏', '失望', '低级', '失败', '一败涂地'}

# ---------------------------------- 清洗函数 ----------------------------------
def clean_text(text):
    text = re.sub(r"[^\u4e00-\u9fa5]", " ", text)
    words = jieba.cut(text)
    return " ".join(w for w in words if w not in stopwords and w.strip())

def emotion_features(text):
    tokens = text.split()
    pos_count = sum(1 for t in tokens if t in positive_words)
    neg_count = sum(1 for t in tokens if t in negative_words)
    return pd.Series([pos_count, neg_count, pos_count - neg_count])

```

```python
# ---------------------------------- 主流程 ----------------------------------
df = pd.read_csv("online_shoping.csv", encoding="utf-8-sig")
df['review'] = df['review'].fillna('').astype(str)

tqdm.pandas()
df['cut_review'] = df['review'].progress_apply(clean_text)
df[['pos_count', 'neg_count', 'emotion_score']] = df['cut_review'].apply(
    emotion_features)

results = []

for cat in sorted(df['cat'].unique()):
    print(f"\n======= 正在处理类别: {cat} =======")
    sub_df = df[df['cat'] == cat]
    if sub_df['label'].nunique() < 2:
        print(f"类别 "{cat}" 不满足二分类要求, 跳过")
        continue

    # 特征提取
    vectorizer = TfidfVectorizer(max_features=8000, ngram_range=(1, 2),
    min_df=2, max_df=0.9)
    X_tfidf = vectorizer.fit_transform(sub_df['cut_review'])
    X_dict = sub_df[['pos_count', 'neg_count', 'emotion_score']]
    X = hstack([X_tfidf, X_dict.values])
    y = sub_df['label']

    # 划分数据集
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y)

    # 训练模型
    start_time = time.time()
    clf = SVC(kernel="linear", C=1, gamma='scale')
    clf.fit(X_train, y_train)
    end_time = time.time()

    # 评估
    y_pred = clf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
```

```
    results.append({'类别': cat, '样本数': len(sub_df), '准确率': f"{acc:.4f}", '训练耗时(s)': f"{end_time - start_time:.2f}"})

    # 保存模型（可选）
    joblib.dump(clf, f"svm_model_{cat}.pkl")
    joblib.dump(vectorizer, f"vectorizer_{cat}.pkl")

# ---------------------------------- 输出准确率表格 ----------------------------------
result_df = pd.DataFrame(results)
print("\n========== 各类别准确率汇总 ==========")
print(result_df.to_markdown(index=False))
```