Modern Linguistics 现代语言学, 2026, 14(4), 10-22
Published Online April 2026 in Hans. https://www.hanspub.org/journal/ml
https://doi.org/10.12677/ml.2026.144261

Hans汉斯

基于IMDb影评的情感分析研究

杨　菊

上海海事大学外国语学院，上海

收稿日期：2026年2月26日；录用日期：2026年3月17日；发布日期：2026年3月31日

摘　要

情感分析的任务是挖掘文本中蕴含的情感倾向，对文本情感极性进行分类，是自然语言处理领域研究的重要问题。本研究通过构建一个情感分析系统，选取IMDb影评数据集作为训练与测试对象，探索如何从数据收集、文本预处理到模型训练的完整流程。同时，针对多语言情感分析的挑战，研究了翻译对情感分类效果的影响。实验表明，基于传统机器学习模型(如朴素贝叶斯和支持向量机)的情感分析方法在高质量预处理和特征提取下能够获得较好的性能，而翻译对分类结果的影响取决于语言的特性和翻译工具的准确性。

关键词

自然语言处理，情感分析，跨语言情感分析，朴素贝叶斯，支持向量机

Sentiment Analysis Research Based on IMDb Movie Reviews

Ju Yang

College of Foreign Language, Shanghai Maritime University, Shanghai

Received: February 26, 2026; accepted: March 17, 2026; published: March 31, 2026

Abstract

The task of sentiment analysis is to mine the emotional tendencies contained in texts and classify the polarity of text emotions, which is an important issue in the field of Natural Language Processing (NLP). This study explores the complete process from data collection and text preprocessing to model training by constructing a sentiment analysis system and selecting the IMDb movie review dataset as the training and testing subject. Simultaneously, addressing the challenges of multilingual sentiment analysis, the study investigates the impact of translation on sentiment classification

文章引用: 杨菊. 基于 IMDb 影评的情感分析研究[J]. 现代语言学, 2026, 14(4): 10-22.
DOI: 10.12677/ml.2026.144261

effectiveness. Experiments show that sentiment analysis methods based on traditional machine learning models (such as Naïve Bayes and Support Vector Machines) can achieve good performance under high-quality preprocessing and feature extraction. The impact of translation on classification results depends on language characteristics and the accuracy of translation tools.

Keywords

Natural Language Processing, Sentiment Analysis, Cross-Lingual Sentiment Analysis, Naïve Bayes, Support Vector Machine

1. 引言

随着互联网的快速发展和普及，大量由用户生成的内容被广泛发布在网络平台上，如电影评论、产品评价以及社交媒体帖子。这些内容不仅承载着用户的情感表达，还蕴含着有价值的信息，对企业决策、产品改进和用户体验优化等方面具有重要影响。情感分析作为自然语言处理(NLP)领域的重要分支，致力于从文本中提取情感信息并进行分类。

近年来，情感分析技术在电子商务、娱乐、公共服务等多个行业得到了广泛应用。例如，在电影行业，通过分析影评平台上的用户评论，电影制作公司可以深入了解观众的情感反馈，进而优化影片的制作和营销策略[1]。然而，尽管情感分析技术在许多领域取得了显著进展，依然面临多语言文本处理、数据质量控制、模型泛化能力等方面的挑战。翻译对文本情感的影响是一个未被充分研究的领域，而理解这种影响对多语言情感分析至关重要[2]。

本研究旨在构建一个完整的情感分析系统，以 IMDb 影评数据集为基础，涵盖从数据收集、文本预处理到模型训练和性能评估的整个流程。研究重点是比较朴素贝叶斯和支持向量机(SVM)两种经典机器学习模型在情感分类任务中的表现，并探索翻译对情感分析效果的影响，为多语言情感分析提供新的思路和实践经验。

2. 文献综述

情感分析中使用的分类器可大致分为机器学习、深度学习与集成学习 3 类。机器学习分类器使用数学模型预测情感；深度学习分类器利用人工神经网络进行情感预测；集成学习方法结合多个分类器实现情感分析[3]。分类器的选择取决于情感分析任务的具体要求与用例。

机器学习方法首先通过预处理与删除任何无关信息的方式标准化文本数据；然后应用 TF-IDF、N-grams 等特征提取技术将文本表示为可输入机器学习分类器的数字特征。机器学习情感分析方法常用的分类器包括支持向量机、朴素贝叶斯、逻辑回归与决策树等(见图 1)。

早期研究多集中于验证基础机器学习算法在情感分析任务上的可行性。例如，Athindran 等[4]利用朴素贝叶斯对某社交媒体平台的客户评论进行分析，取得了 77%的准确率，证明了该方法在特定场景下的有效性。然而，此类研究通常基于规模有限、领域特定的数据集(如电商或社交媒体评论)，其结论的泛化能力存在局限。Vanaja 与 Belwal [5]在亚马逊产品评论上对比了朴素贝叶斯与支持向量机(SVM)，发现前者准确率(90.42%)高于后者(83.42%)。这一矛盾性结果提示，模型性能高度依赖于数据特征、预处理流程

及任务定义，而非算法本身存在绝对的优劣。Iqbal 等人的研究[6]进一步表明，通过精心设计特征组合(如结合 N-gram)，最大熵模型在 IMDb 等标准数据集上能达到 88%以上的准确率。这些研究虽然展示了传统机器学习方法的潜力，但普遍存在对上下文语义理解不足、对跨语言或翻译文本处理研究缺失等问题。

因此，本研究选取大规模的 IMDb 影评数据集，系统比较 NB 与 SVM 模型，并着重探讨翻译对模型性能的影响，旨在弥补现有研究在跨语言情感分析泛化性探讨方面的不足。

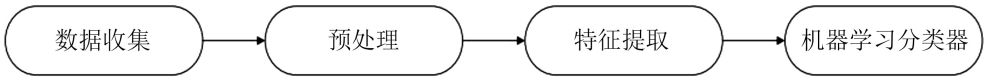

Figure 1. The process of sentiment analysis using machine learning methods
图 1. 机器学习方法情感分析过程

3. 相关技术

3.1. IMDb 影评数据集

本研究选择 IMDb 影评数据集，包含大量标注为“正面”或“负面”的影评。数据集包括评论与情感两列，评论融合了故事情节与个人观点，其语言复杂性增加了情感分析的难度。这些数据以文本形式存储，标签为二分类，有利于构建情感分类模型。

3.2. 朴素贝叶斯(NB)

朴素贝叶斯是基于贝叶斯定理的分类算法，其假设特征之间相互独立，使得计算条件概率更加简化，特别适用于文本分类等领域，在处理大规模数据集与高维特征空间时具有高效性[7]。朴素贝叶斯分类模型是一种常见的分类模型，其基本结构如图 2 所示，其中，C 表示分类类别，x 表示各特征属性。设样本集合 $N=\{N_1,N_2,\cdots,N_n\}$，样本的特征属性集合 X 可设为 $\{x_1,x_2,\cdots,x_n\}$，类别集合 $Y=\{y_1,y_2,\cdots,y_n\}$，可得到

$$P(y_i|,x_2,\cdots,x_m)=\frac{P(y_i)\prod_{k=1}^{m}P(x_k|)}{P(X)}$$

由于 P(X)为固定值，因此，在比较后检验概率的大小时，只需考虑分子的大小即可：

$$Y_{max}=\arg\max_{y_i}P(y_i|,x_2,\cdots,x_m)=\arg\max_{y_i}P(y_i)\prod_{k=1}^{m}P(x_k|)$$

3.3. 支持向量机(SVM)

SVM 是有监督学习算法的一种，可以用来解决许多关键问题，比如模式识别领域中的数据分类问题，SVM 要解决的问题一般用一个经典的二分类问题来描述[8]。其中，支持向量机通过寻找最优超平面来实现数据点的二分类，这个超平面被设计为最大化两个类别的支持向量到超平面的距离，从而提高模型的泛化能力。支持向量机在处理线性可分问题时表现出色，同时也能通过核函数处理非线性问题。

4. 操作步骤

4.1. 数据收集

进入 IMDb 官网下载数据集，共 50,000 条影评，其中 25,000 作为训练集，25,000 作为测试集，并使用 Google Translate 将影评翻译为中文。

4.2. 文本预处理

文本预处理是自然语言处理(NPL)中的关键步骤，对于情感分析而言尤为重要。

4.2.1. 数据清洗

英文文本

首先查看原始文本(见图 2)：

```
print("原始文本:")
print(text)
```

```
原始文本:
Oh, brother...after hearing about this  ridiculous film for umpteen years all I can think of is that old Peggy Lee song..<br /><br />https://www.japan-travel.cn/vj/food/. "Is that all there is??" ...I was just an early teen when this smoked fish hit the U.S. I was too young to get in the theater (although I did manage to sneak into "Goodbye Columbus"). Then a screening at a local film museum beckoned - Finally I could see this film, except now I was as old as my parents were when they schlepped to see it!!<br /><br />😕🙁The ONLY reason this film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its U.S. release.
```

Figure 2. Original English text
图 2. 原始文本

URL 是通常在文本分析中不需要的内容，尤其是当文本关注的是内容和情感时，链接并不提供任何语义信息，因此需要去除文本中的 URL。HTML 标签通常用于网页的布局，在文本分析中往往也没有实质性的内容。

通过正则表达式去除了文本中的 HTML 标签。<.*?>：匹配所有以<开头，>结尾的内容。http\S+：匹配以 http 开头的 URL，S+表示匹配所有非空白字符。|是正则表达式中的“或”操作符，表示匹配 http 或 www 或 https 开头的字符串。https\S+和 www\S+：匹配以 https 或 www 开头的 URL。清理掉了无关的超链接信息，减少了文本中的杂项内容。成功地去掉了文本中的
等标签，使得文本内容更加简洁(见图 3)。

```
# 去除URL,HTML
text_no_html = re.sub(r'<.*?>', '', text)
text_no_url = re.sub(r'http\S+|www\S+|https\S+', '', text_no_html)
print("\n去除HTML标签后的文本:")
print(text_no_html)
```

```
去除HTML标签后的文本:
Oh, brother...after hearing about this  ridiculous film for umpteen years all I can think of is that old Peggy Lee song..https://www.japan-travel.cn/vj/food/. "Is that all there is??" ...I was just an early teen when this smoked fish hit the U.S. I was too young to get in the theater (although I did manage to sneak into "Goodbye Columbus"). Then a screening at a local film museum beckoned - Finally I could see this film, except now I was as old as my parents were when they schlepped to see it!!😕🙁The ONLY reason this film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its U.S. release.
```

Figure 3. Removal of URLs and HTML tags
图 3. 去除 URL 和 HTML

```
# 去除标点符号
text_no_punctuation = re.sub(r'[^\w\s]', '', text_no_url)
print("\n去除标点符号后的文本:")
print(text_no_punctuation)
```

```
去除标点符号后的文本:
Oh brotherafter hearing about this  ridiculous film for umpteen years all I can think of is that old Peggy Lee song Is that all there is I was just an early teen when this smoked fish hit the US I was too young to get in the theater although I did manage to sneak into Goodbye Columbus Then a screening at a local film museum beckoned  Finally I could see this film except now I was as old as my parents were when they schlepped to see itThe ONLY reason this film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its US release
```

Figure 4. Removal of punctuation and emoticons
图 4. 去除标点符号

标点符号在一些任务中可能没有语义上的帮助，尤其是当任务关注的是词汇或情感时。表情符号通常无法被编码，会引入错误，所以需要去除标点符号和表情符号。使用正则表达式来匹配并去除标点符号，表情符号。\w：匹配字母、数字和下划线。\s：匹配空格、换行符等空白字符。[,.!?]：保留常见的标点符号，避免删除必要的标点。文本变得更加简洁，去掉了表情符号、句号、逗号、问号等，这对于后续的模型训练会有帮助(见图 4)。

为了统一文本的格式，需要将所有字符转换为小写。这样，大小写不一致的词汇会被认为是同一个词。使用 Python 字符串的 lower()方法将文本转换为小写。\s+：匹配一个或多个空白字符。' '：将匹配到的多个空白字符替换为一个空格。strip()：去除文本两端的空白字符。通过转换为小写，消除了大小写带来的不一致性，为后续的分析提供了更加统一的文本数据(见图 5)。

```
# 转换为小写
text_lower = text_no_punctuation.lower()
print("\n转换为小写后的文本:")
print(text_lower)
```

```
转换为小写后的文本:
oh brotherafter hearing about this  ridiculous film for umpteen years all i can think of is that old peggy lee song is that all there is i was just an
early teen when this smoked fish hit the us i was too young to get in the theater although i did manage to sneak into goodbye columbus then a screening
at a local film museum beckoned  finally i could see this film except now i was as old as my parents were when they schlepped to see itthe only reason t
his film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its us release
```

Figure 5. Converted to lowercase
图 5. 转换为小写

文本中的多余空格会影响文本的处理效果，因此需要去除。使用正则表达式将多个连续空格替换为单个空格，并去掉两端的空格。去除多余空格后，文本格式变得更加规整，不再包含多个连续的空格(见图 6)。

```
# 去除多余空格
text_no_extra_spaces = re.sub(r'\s+', ' ', text_lower).strip()
print("\n去除多余空格后的文本:")
print(text_no_extra_spaces)
```

```
去除多余空格后的文本:
oh brotherafter hearing about this ridiculous film for umpteen years all i can think of is that old peggy lee song is that all there is i was just an ea
rly teen when this smoked fish hit the us i was too young to get in the theater although i did manage to sneak into goodbye columbus then a screening at
a local film museum beckoned finally i could see this film except now i was as old as my parents were when they schlepped to see itthe only reason this
film was not condemned to the anonymous sands of time was because of the obscenity case sparked by its us release
```

Figure 6. Removal of extra spaces
图 6. 去除多余空格

分词是文本处理中的重要步骤。它将文本切分为一个个单独的词，为后续的分析、特征提取或建模做好准备。需要使用 nltk 库中的 word_tokenize 进行分词(见图 7)。

```
# 分词
tokens = word_tokenize(text_no_extra_spaces)
print("\n分词后的文本:")
print(tokens)
```

```
分词后的文本:
['oh', 'brotherafter', 'hearing', 'about', 'this', 'ridiculous', 'film', 'for', 'umpteen', 'years', 'all', 'i', 'can', 'think', 'of', 'is', 'that', 'ol
d', 'peggy', 'lee', 'song', 'is', 'that', 'all', 'there', 'is', 'i', 'was', 'just', 'an', 'early', 'teen', 'when', 'this', 'smoked', 'fish', 'hit', 'th
e', 'us', 'i', 'was', 'too', 'young', 'to', 'get', 'in', 'the', 'theater', 'although', 'i', 'did', 'manage', 'to', 'sneak', 'into', 'goodbye', 'columbu
s', 'then', 'a', 'screening', 'at', 'a', 'local', 'film', 'museum', 'beckoned', 'finally', 'i', 'could', 'see', 'this', 'film', 'except', 'now', 'i', 'w
as', 'as', 'old', 'as', 'my', 'parents', 'were', 'when', 'they', 'schlepped', 'to', 'see', 'itthe', 'only', 'reason', 'this', 'film', 'was', 'not', 'con
demned', 'to', 'the', 'anonymous', 'sands', 'of', 'time', 'was', 'because', 'of', 'the', 'obscenity', 'case', 'sparked', 'by', 'its', 'us', 'release']
```

Figure 7. Word segmentation
图 7. 分词

停用词是指那些对文本分析没有实际帮助的词，如“the”、“is”、“in”等。去除停用词有助于减少分析的噪音，提高处理效率。使用 nltk 提供的停用词列表来进行过滤(见图 8)。

```
# 去除停用词
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
print("\n去除停用词后的文本:")
print(filtered_tokens)
```

```
去除停用词后的文本:
['oh', 'brotherafter', 'hearing', 'ridiculous', 'film', 'umpteen', 'years', 'think', 'old', 'peggy', 'lee', 'song', 'early', 'teen', 'smoked', 'fish',
'hit', 'us', 'young', 'get', 'theater', 'although', 'manage', 'sneak', 'goodbye', 'columbus', 'screening', 'local', 'film', 'museum', 'beckoned', 'final
ly', 'could', 'see', 'film', 'except', 'old', 'parents', 'schlepped', 'see', 'itthe', 'reason', 'film', 'condemned', 'anonymous', 'sands', 'time', 'obsc
enity', 'case', 'sparked', 'us', 'release']
```

Figure 8. Removal of stopwords
图 8. 去除停用词

最后，将处理后的词汇拼接回文本(见图 9)：

```
# 拼接文本
processed_text = ' '.join(filtered_tokens)
print("\n处理后的最终文本:")
print(processed_text)
```

```
处理后的最终文本:
oh brotherafter hearing ridiculous film umpteen years think old peggy lee song early teen smoked fish hit us young get theater although manage sneak goo
dbye columbus screening local film museum beckoned finally could see film except old parents schlepped see itthe reason film condemned anonymous sands t
ime obscenity case sparked us release
```

Figure 9. Text concatenation
图 9. 拼接文本

通过上述步骤，对原始文本进行了完整的清洗流程，包括去除 HTML 标签、URL、表情符号、标点符号，统一大小写，去除多余空格，并最终分词和去除停用词。每一步都确保了数据的清洁和统一，为后续的文本分析或机器学习任务做好了准备。

用 Google Translation 将文本翻译为中文，进行数据清洗中文文本(见图 10)：

```
text = """值得租借，尤其是如果你喜欢动作片的话。这部电影有常见的汽车追逐、范·达姆踢腿式打斗、40 发霰弹枪射击战，甚至还有恐怖分子式炸弹。所有这些都很有趣，处理得当，但
print("原始文本：")
print(text)
```

Figure 10. Original Chinese text
图 10. 中文原始文本

去除特殊字符和表情符号(见图 11)：

```
# 去掉特殊字符和表情符号
cleaned_text_step1 = re.sub(r'[^\w\s\u4e00-\u9fa5]', '', text)
print("去除特殊字符,表情后的文本：")
print(cleaned_text_step1)
```

Figure 11. Removal of special characters and emoticons
图 11. 去除特殊字符和表情符号

去除链接(见图 12)：

```
cleaned_text_step1 = re.sub(r'http\S+|www\S+|https\S+', '', cleaned_text_step1)  # 去掉链接
print("去除链接后的文本：")
print(cleaned_text_step1)
```

```
去除链接后的文本：
值得租借尤其是如果你喜欢动作片的话这部电影有常见的汽车追逐范达姆踢腿式打斗40 发霰弹枪射击战甚至还有恐怖分子式炸弹所有这些都很有趣处理得当但如果你以前看过没有什么能真正让你震惊的 1940 年代好莱坞电影中墨西哥人的定型角色所有人的表演都还过得去但同样没有什么特别之处我认为主要的反派塑造得很好演得也相当好到电影结束时你肯定知道谁是好人谁不是当真正坏人得到应有的惩罚时情绪得到了提升非常简单但你没想到哈姆雷特对吧我发现唯一真正令人讨厌的是最后一场打斗场景中不断切换到 VD 的女儿还不错不好还过得去 4
```

Figure 12. Removal of links
图 12. 去除链接

中文分词指的是将一个汉字序列切分成一个个单独的词，分词的过程实际上就是将连续、完整的汉字序列，按照一定的方法，组合成词序列的过程[9]。中文分词的实现，一般均是在程序中直接调用某个成品分词系统的接口函数，jieba 目前被认为是最好用的基于 Python 实现的分词系统，很容易就可以实现分词调用和词性标注，可以一定程度上实现的未登录词识别，由于中文缺乏空格分隔符，需依赖分词算法。还能通过自建词库扩展实现新词的登录。

使用 jieba.lcut()进行分词，以便对其进行分析。输出为列表形式，每个元素是一个分词单元(见图 13)。

```
segmented_words = jieba.lcut(cleaned_text_step1)
print("分词结果：")
print(segmented_words)
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\77599\AppData\Local\Temp\jieba.cache
Loading model cost 0.267 seconds.
Prefix dict has been built successfully.
分词结果：
['值得', '租借', '尤其', '是', '如果', '你', '喜欢', '动作片', '的话', '这部', '电影', '有', '常见', '的', '汽车', '追逐', '范', '达姆', '踢腿', '式', '打斗', '40', ' ', '发', '霰弹枪', '射击', '战', '甚至', '还有', '恐怖', '分子式', '炸弹', '所有', '这些', '都', '很', '有趣', '处理', '得当', '但', '如果', '你', '以前', '看过', '没有', '什么', '能', '真正', '让', '你', '震惊', '的', ' ', '1940', ' ', '年代', '好莱坞', '电影', '中', '墨西哥人', '的', '定型', '角色', '所有人', '的', '表演', '都', '还过得去', '但', '同样', '没有', '什么', '特别之处', '我', '认为', '主要', '的', '反派', '塑造', '得', '很', '好', '演得', '也', '相当', '好', '到', '电影', '结束', '时', '你', '肯定', '知道', '谁', '是', '好人', '谁', '不是', '当', '真正', '坏人', '得到', '应有', '的', '惩罚', '时', '情绪', '得到', '了', '提升', '非常简单', '但', '你', '没想到', '哈姆雷特', '对', '吧', '我', '发现', '唯一', '真正', '令人讨厌', '的', '是', '最后', '一场', '打斗', '场景', '中', '不断', '切换', '到', ' ', 'VD', ' ', '的', '女儿', '还', '不错', '不好', '还过得去', ' ', '4']
```

Figure 13. Word segmentation
图 13. 中文分词

删除语义贡献较小的常用词，如“的”“了”“是”，以突出文本中的关键信息。这样可以降低冗余信息对分析的干扰，提升分析结果的有效性。使用停用词表(stopword.txt)，里面包含一组需删除的高频无意义词。使用 set 函数用于快速查询停用词(见图 14)。

```
stopwords = set()
with open('stopword.txt', 'r', encoding='utf-8') as f:
    for line in f:
        stopwords.add(line.strip())

# 去停用词
filtered_words = [word for word in segmented_words if word not in stopwords and word.strip()]
print("去停用词后的分词结果：")
print( filtered_words)
```

```
去停用词后的分词结果：
['值得', '租借', '喜欢', '动作片', '这部', '电影', '常见', '汽车', '追逐', '范', '达姆', '踢腿', '式', '打斗', '40', '发', '霰弹枪', '射击', '战', '恐怖', '分子式', '炸弹', '有趣', '得当', '看过', '震惊', '1940', '年代', '好莱坞', '电影', '墨西哥人', '定型', '角色', '所有人', '表演', '还过得去', '特别之处', '反派', '塑造', '演得', '电影', '结束', '时', '肯定', '好人', '坏人', '应有', '惩罚', '时', '情绪', '提升', '非常简单', '没想到', '哈姆雷特', '发现', '唯一', '令人讨厌', '一场', '打斗', '场景', '切换', 'VD', '女儿', '不错', '不好', '还过得去']
```

Figure 14. Remove stop words
图 14. 删除停用词

4.2.2. 特征提取

英文

采用 TF-IDF 向量化影评文本，生成稀疏特征矩阵供模型训练。TF-IDF 是一种用于评估文本中单词重要性的统计方法。它可以衡量一个词在某个文档中出现的频率，同时考虑到这个词在所有文档中的分布情况，从而帮助找到对文档主题有代表性的关键词[10]。

首先使用 TfidfVectorizer 中的 get_feature_names_out 方法查看所有的特征词(见图 15)：

```
feature_names = vectorizer.get_feature_names_out()
print("所有特征词：")
print(feature_names)
```

```
所有特征词：
['10' '100' '1010' '11' '110' '12' '13' '13th' '14' '15' '16' '17' '18'
 '1930s' '1933' '1940s' '1950s' '1960s' '1968' '1970s' '1971' '1972'
 '1973' '1980' '1980s' '1983' '1984' '1990' '1990s' '1996' '1999' '1st'
 '20' '2000' '2001' '2002' '2003' '2004' '2005' '2006' '2007' '20th' '210'
 '24' '25' '2nd' '30' '3000' '30s' '310' '35' '3d' '3rd' '40' '40s' '410'
 '45' '50' '50s' '60' '60s' '70' '70s' '710' '75' '80' '80s' '810' '90'
 '90s' '910' '911' 'abandoned' 'abc' 'abilities' 'ability' 'able'
 'aboutbr' 'abraham' 'absence' 'absent' 'absolute' 'absolutely' 'absurd'
 'abuse' 'abused' 'abusive' 'abysmal' 'academy' 'accent' 'accents'
 'accept' 'acceptable' 'accepted' 'access' 'accident' 'accidentally'
 'accompanied' 'accomplished' 'according' 'account' 'accuracy' 'accurate'
 'accused' 'achieve' 'achieved' 'achievement' 'acid' 'act' 'acted'
 'acting' 'actingbr' 'action' 'actions' 'activities' 'actor' 'actors'
 'actress' 'actresses' 'acts' 'actual' 'actually' 'ad' 'adam' 'adams'
 'adaptation' 'adaptations' 'adapted' 'add' 'added' 'adding' 'addition'
 'additional' 'adds' 'adequate' 'admire' 'admit' 'admittedly' 'adopted'
 'adorable' 'adult' 'adults' 'advance' 'advanced' 'advantage' 'adventure'
```

Figure 15. View feature words
图 15. 查看特征词

然后使用 TfidfVectorizer 的 fit_transform 方法生成稀疏矩阵元素，将稀疏矩阵元素转化为密集数组，方便查看特征向量(见图 16)。

```
# 将稀疏矩阵转换为稠密矩阵
dense_X_train = X_train.toarray()
print(dense_X_train[0])
```

```
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.17069147 0.
0.         0.         0.         0.         0.1261259  0.
0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.         0.
```

Figure 16. Convert the sparse matrix to a dense matrix
图 16. 将稀疏矩阵转换为稠密矩阵

发现有很多为 0 的 TF-IDF 值，使用 nonzero()函数去除 0 值，打印所有非零特征词的 TF-IDF 值(见图 17)。

```
#查看非零的
for index in X_train[0].nonzero()[1]:
    print(f"{feature_names[index]}: {dense_X_train[0][index]}")
```

```
rented: 0.09098604945607955
video: 0.06807156872742375
store: 0.08618900360932179
surrounded: 0.10626261133905913
released: 0.07431919347789956
heard: 0.07168415184460823
tried: 0.07808704678883224
enter: 0.10120131188167492
country: 0.07772320432276826
fan: 0.06368824784025216
films: 0.04428740437476804
considered: 0.17091120389969938
controversial: 0.10665135766198437
really: 0.11200094479249881
br: 0.07837991838054909
plot: 0.08988819951592658
young: 0.05501364527129411
swedish: 0.3508729832238129
drama: 0.1411205474237563
student: 0.09217658045752446
named: 0.0788645452375216
lena: 0.12590347261233328
wants: 0.13970039959978645
learn: 0.0797331458947631
life: 0.047244940132955944
particular: 0.07838382017879651
focus: 0.0853426337477634
making: 0.05785150218388594
sort: 0.06812613821543437
documentary: 0.08105207148614323
average: 0.08008786007315331
thought: 0.054313003795587775
certain: 0.07796490068822542
political: 0.08541786260315345
issues: 0.17984465955960274
vietnam: 0.1077335821485219
war: 0.07087847732085614
race: 0.09510478992682739
united: 0.09948236561313493
states: 0.09184355406397651
asking: 0.09745631110450896
ordinary: 0.09620448167073031
opinions: 0.11450332987148461
politics: 0.10110638512881769
sex: 0.34902451271542584
teacher: 0.09627562711049094
```

Figure 17. TF-IDF of non-zero feature words
图 17. 非零特征词的 TF-IDF

中文

使用词袋模型进行特征提取，将文本视为一个单词袋，忽略单词的顺序和语法结构，仅关注单词的出现频率。每个文本被表示为一个固定大小的向量，其中每个位置对应一个特定单词的出现次数。

使用 DataFrame 将词频矩阵可视化(见图 18)。

```
# 提取特征名称和词频矩阵
feature_names = vectorizer.get_feature_names_out()
word_counts = X_train_vec.toarray()
word_counts_df = pd.DataFrame(word_counts, columns=feature_names)
print("词频矩阵：")
print(word_counts_df.head())
```

```
词频矩阵：
   00  000000000000001  000001  00000110  000015  0001  001  0010  002  \
0   0                0       0         0       0     0    0     0    0
1   0                0       0         0       0     0    0     0    0
2   0                0       0         0       0     0    0     0    0
3   0                0       0         0       0     0    0     0    0
4   0                0       0         0       0     0    0     0    0

   00383042  ...  龙虎风云  龙虾  龙蛇  龙蛋龙  龙门客栈  龙马  龙骑兵  龙骑士  龟人  龟界
0         0  ...     0   0   0    0     0   0    0    0   0   0
1         0  ...     0   0   0    0     0   0    0    0   0   0
2         0  ...     0   0   0    0     0   0    0    0   0   0
3         0  ...     0   0   0    0     0   0    0    0   0   0
4         0  ...     0   0   0    0     0   0    0    0   0   0

[5 rows x 87938 columns]
```

Figure 18. Word frequency martix
图 18. 词频矩阵

4.3. 模型训练与测试

4.3.1. 朴素贝叶斯(NB)模型训练

训练一个朴素贝叶斯分类器来对文本进行分类(见图 19)。朴素贝叶斯通过计算先验概率和条件概率，推断文本类别。假设词汇独立性，其分类效果依赖于特征质量。MultinomialNB()是 sklearn.naive_bayes 模块的一个类，它实现了多项式朴素贝叶斯分类器，通常用于文本分类任务。该分类器假设特征是独立的，并且服从多项式分布。

```
# 朴素贝叶斯
from sklearn.naive_bayes import MultinomialNB
nb_model = MultinomialNB()
```

```
nb_model.fit(X_train, train_labels)
```

```
MultinomialNB()
```

```
test_predictions = nb_model.predict(X_test)
accuracy = accuracy_score(test_labels, test_predictions)
print(f"Naive Bayes Accuracy: {accuracy:.4f}")
```

```
Naive Bayes Accuracy: 0.8437
```

Figure 19. Naive bayes training
图 19. 朴素贝叶斯训练

fit()方法用于训练 SVM 模型。它接收训练数据的特征矩阵和训练数据的标签。通过寻找最优的超平面来拟合训练数据，并确定每个类别的决策边界。模型学习数据中的模式，用于对新的数据进行预测。

predict()方法用于对新的输入数据进行分类。该方法返回一个类别标签的预测结果列表。目标是使用训练好的模型对测试数据进行预测，并根据这些预测与真实标签进行比较，以评估模型的性能。

accuracy_score()是一个评估模型准确度的函数，来自 sklearn.metrics 模块。它计算模型预测结果与实际标签之间的匹配程度，并返回一个准确率值，表示分类器的性能。准确率是分类正确的样本占总样本数的比例。

4.3.2. 支持向量机(SVM)模型训练

训练一个 SVM 模型，自动学习如何将不同类别分开。它通过找到决策边界，将数据点分类。SVM 通过寻找最大间隔超平面分类数据，适合处理高维稀疏数据。

使用 SVM 进行训练并预测代码及结果(见图 20)。

```
# SVM训练
svm_model = LinearSVC()

#训练
svm_model.fit(X_train, train_labels)
```

```
LinearSVC()
```

```
test_predictions = svm_model.predict(X_test)
accuracy = accuracy_score(test_labels, test_predictions)
print(f"SVM Accuracy: {accuracy:.4f}")
```

```
SVM Accuracy: 0.8617
```

Figure 20. Support vectot machine
图 20. 支持向量机训练

5. 实验与结果

5.1. 实验结果

5.1.1. 朴素贝叶斯与支持向量机

模型	数据集	准确率 (Accuracy)	召回率－正类 (Recall-Positive)	召回率－负类 (Recall-Negative)	F1 分数 (F1-Score)
朴素贝叶斯	英文(EN)	84.37%	0.8721	0.8152	0.8612
支持向量机	英文(EN)	86.17%	0.8450	0.8785	0.8413
朴素贝叶斯	中文(ZH)	82.37%	0.8510	0.7965	0.8320
支持向量机	中文(ZH)	83.16%	0.8201	0.8432	0.8189

根据获得数据可见，支持向量机在准确率上略高于朴素贝叶斯，表明在整体分类任务中，SVM 对数据的分类效果稍好，能够更好地预测情感标签。召回率反映了模型对正类(例如积极情感或消极情感)的识别能力，即在所有实际为正类的样本中，模型正确识别的比例。朴素贝叶斯略好于 SVM，表明其在识别正类情感方面更出色。F1 分数综合考虑了模型的精确度和召回率，朴素贝叶斯在 F1 分数上表现略优于

支持向量机，表明其在保持较高召回率的同时，也能够相对保持较高的精确度，从而在综合性能上略胜一筹。

因此，如果情感类别不平衡或者需要更高的正类识别能力，朴素贝叶斯可能会更有优势；而如果任务偏向于整体准确性，支持向量机则可能表现更好。

5.2.2. 翻译对分类性能的影响

总体来看，翻译后的中文数据比原始英文数据的分类准确率低。朴素贝叶斯和支持向量机的准确率都在中文翻译版本上有所下降，说明翻译过程可能引入了额外的噪声，导致模型在理解和分类情感时产生了一些偏差。

翻译会显著改变文本所携带的感情。原文的正面或负面情感在翻译后经常变得中性，在机器翻译中，这种现象更为明显。支持向量机在翻译后的中文数据中表现相对更好。支持向量机可能对语义信息和特征更敏感，因此在翻译过程中丢失的部分信息影响较小，而朴素贝叶斯的表现可能更依赖于原始数据中的词频和统计模式，翻译可能更容易影响其分类效果。同时，翻译还可能影响模型的泛化能力。

6. 结论与未来工作

本研究通过比较朴素贝叶斯和支持向量机(SVM)在情感分类任务中的表现，揭示了两种经典机器学习模型的优劣势。结果表明，SVM 在整体准确率上优于朴素贝叶斯，能够更好地进行情感标签预测。然而，朴素贝叶斯在召回率和 F1 分数上略优，尤其在情感类别不平衡的情况下，能够较好地识别正类情感。因此，对于需要高召回率或更好正类识别的任务，朴素贝叶斯可能更为适用；而当任务侧重于整体分类准确性时，支持向量机则显示出更为优越的表现。

在多语言情感分析的实验中，翻译对分类性能的影响不容忽视。翻译的质量是影响情感一致性的关键因素。常见问题包括情感词的缺失和替换；翻译不准确导致情感发生变化；翻译文本中否定表达的位置或形式被误解。SVM 在翻译后的中文数据中表现相对更好，可能由于其对语义信息和特征的敏感性较高，而朴素贝叶斯则更依赖于原始数据的词频和统计模式，受到翻译影响较大。这一发现强调了翻译质量和语义差异对情感分析模型性能的重要性。

未来的工作可以着重于进一步优化模型在多语言环境下的适应性。首先，针对翻译带来的信息损失，可以采用更精确的情感词典对齐或训练针对不同语言的情感分析模型。其次，探索深度学习等更复杂模型在多语言情感分析中的表现，可能有助于提高翻译后数据的处理能力。此外，考虑到情感分析中的语言和文化差异，未来的研究还可以从跨文化情感分析的角度出发，进一步探索如何减轻翻译对模型泛化能力的负面影响，从而提升情感分析系统的准确性。

参考文献

[1] 刘玲玉, 邓燕燕. 基于 Python 情感分析和批评隐喻的网络话语分析——以影片《流浪地球》中美德影评为例[J]. 江苏大学学报(社会科学版), 2022, 24(3): 76-88.

[2] 徐月梅, 曹晗, 王文清, 等. 跨语言情感分析研究综述[J]. 数据分析与知识发现, 2023, 7(1): 1-21.

[3] 王钦炀, 施水才, 王洪俊. 文本情感分析综述[J/OL]. 软件导刊: 1-10. https://kns.cnki.net/kcms2/article/abstract?v=9oxawJFDgQVlbfQPmNjR6YQ2RoYUdt8npuN-BmgH4VacHCNBuKrECb5UR4dt_k6ff_Hb1Jo3J33gxxgnwmnxKSjZoTEqFRB3ZJg2zCAfZt1wm-H_FK23ETF249ZBAMpX7Zdn1IjDGd4n0oE_Y4wvoubfDOM4Age8O0yuyJbM9R9w=&uniplatform=NZKPT, 2024-12-23.

[4] Srivats Athindran, N., Manikandaraj, S. and Kamaleshwar, R. (2018) Comparative Analysis of Customer Sentiments on Competing Brands Using Hybrid Model Approach. 2018 3*rd International Conference on Inventive Computation Technologies* (*ICICT*), Coimbatore, 15-16 November 2018, 348-353. https://doi.org/10.1109/icict43934.2018.9034283

[5] Vanaja, S. and Belwal, M. (2018) Aspect-Level Sentiment Analysis on E-Commerce Data. 2018 *International Conference on Inventive Research in Computing Applications* (*ICIRCA*), Coimbatore, 11-12 July 2018, 1275-1279. https://doi.org/10.1109/icirca.2018.8597286

[6] Iqbal, N., Chowdhury, A.M. and Ahsan, T. (2018) Enhancing the Performance of Sentiment Analysis by Using Different Feature Combinations. 2018 *International Conference on Computer*, *Communication*, *Chemical*, *Material and Electronic Engineering* (*IC*4*ME*2), Rajshahi, 8-9 February 2018, 1-4. https://doi.org/10.1109/ic4me2.2018.8465673

[7] 游棉州. 情感分析的算法与技术应用[J]. 电子技术, 2022, 51(9): 190-191.

[8] 陈琪, 张莉, 蒋竞, 等. 一种基于支持向量机和主题模型的评论分析方法[J]. 软件学报, 2019, 30(5): 1547-1560.

[9] 张小艳, 白瑜. 基于加权融合字词向量的中文在线评论情感分析[J]. 计算机应用研究, 2022, 39(1): 31-36.

[10] 刘祉燊, 张倩, 周菠, 等. 基于支持向量机的中文文本情感分析方法研究[J]. 科技创新与应用, 2022, 12(32): 27-30.