华中科技大学计算机科学与技术学院

《自然语言处理》结课报告

题目：基于五种方法电影评论分类

专　　业	计算机科学与技术
班　　级	计　科 2xxx
学　　号	Uxxxxxxx
姓　　名	xxxxx
成　　绩	
指导教师	辜希武
时　　间	2025 年 7 月 1 日

目录

摘要

本文面向自然语言处理课程，构建了一个统一的实验框架，旨在对比评估多种经典机器学习模型及神经网络在**英文电影评论情感分类**任务上的性能。实验采用自建的包含约 1000 条英文电影评论的数据集，评论被清晰地标注为正面、负面或中性三种情感（为进行二元分类，实验中主要针对正面与负面情感，中性评论被预先滤除）。所对比评估的方法包括基于**词频-逆文档频率（TF-IDF）**特征的传统机器学习模型：**逻辑回归（Logistic Regression）**、**朴素贝叶斯（Naive Bayes）**、**支持向量机（SVM）**和**随机森林（Random Forest）**，以及一个同样基于 TF-IDF 特征输入的**前馈神经网络（Feed-forward Neural Network）**。在统一的 Python 脚本环境中，完成了数据自动加载、文本预处理（包括小写化、去除标点、分词、去除停用词）、特征提取、多模型批量训练、交叉验证（隐式通过 train-test split 的 randomstate 控制可复现划分）及性能评估。主要考察并计算了各模型在测试集上的*准确率*（*Accuracy*）、*精确率*（*Precision*）、*召回率*（*Recall*）和 *F1 值*。实验结果初步显示：基于 TF-IDF 特征的**逻辑回归**与**支持向量机**在 F1 指标上通常表现出较好的平衡性与分类鲁棒性；**朴素贝叶斯**模型展现了显著的训练速度优势；而**随机森林**在适当调参后能提供较强的性能。所构建的**前馈神经网络**在当前数据集规模和特征表示下，其性能与经过优化的传统模型相当，这提示了其在更大规模数据集和更复杂特征工程（如词嵌入）上的应用潜力。此外，该框架还集成了一个用户交互模块，允许用户输入自定义文本，并由已训练模型即时预测其情感倾向。初步的误差分析指向了若干影响分类准确性的因素，例如讽刺与反讽表达的识别困难、复杂句式与上下文依赖、否定结构的正确处理、以及情感词典覆盖不足或情感强度表达的细微差异等。最后，报告探讨了未来可能的改进方向，包括引入**词嵌入（如 Word2Vec、GloVe）**作为更丰富的语义特征、采用更先进的序列模型如**循环神经网络（RNN）**、**长短期记忆网络（LSTM）**或 **Transformer 模型（如 BERT）**进行端到端的学习，以及探索更细致的**方面级情感分析（ABSA）**，从而提升模型对复杂文本情感的理解与分类能力。本报告提供了完整的代码实现与所用数据集，以支持实验的可复现性与进一步研究。

关键词：情感分析， 文本分类， TF-IDF， 机器学习， 神经网络， 电影评论， 性能评估

1 绪论

1.1 研究背景与动机

随着互联网的飞速发展和社交媒体的普及，用户生成的文本数据（如产品评论、社交动态、博客文章等）呈爆炸式增长。从这些海量文本中自动分析和提取作者的主观意见、情感色彩和态度，即**情感分析（Sentiment Analysis）**或称**观点挖掘（Opinion Mining）**，已成为**自然语言处理（NLP）**领域中一个至关重要的研究方向。电影评论作为一种典型的用户生成内容，蕴含了观众对电影各方面的丰富情感表达。准确地对电影评论进行情感分类，不仅能够帮助潜在观众做出观影选择，也能为电影制作方和发行方提供有价值的市场反馈。因此，系统性地比较不同情感分析方法在电影评论数据集上的表现，对于理解各算法的特性、推动技术在实际场景中的应用具有重要意义。

1.2 情感分析的研究意义

情感分析技术在学术研究与商业应用中均具有广泛且深远的价值：

1. **商业智能与市场决策**——企业可以通过分析用户对产品（如电影、商品、服务）的评论情感，洞察消费者偏好，优化产品设计，提升服务质量，并制定更精准的营销策略和品牌声誉管理方案；

2. **舆情监控与社会治理**——政府和相关机构能够实时追踪公众对社会事件、政策法规的情感倾向，及时发现潜在风险，为公共关系管理和应急响应提供数据支持；

3. **个性化推荐系统**——通过理解用户对不同项目（如电影、音乐、新闻）的情感反馈，推荐系统可以提供更符合用户个性化口味的内容，从而提升用户体验和平台粘性；

4. **人机交互与智能客服**——赋予机器理解和响应用户情感的能力，可以使对话系统、虚拟助手等更加智能和人性化，提升交互的自然度和用户满意度。

1.3 国内外研究现状

当前，情感分析的主流研究方法大致可以归纳为以下几类：

- **基于情感词典的方法**：此类方法依赖预先构建的情感词典（包含词语及其情感极性与强度），通过匹配文本中的情感词并聚合其得分来判断整体情感。优点是简单直观，但对词典的完备性、领域适应性以及复杂语言现象（如反讽、否定）的处理能力有限。

- **传统的机器学习方法**：将情感分析视为文本分类任务，利用如词袋模型（*BoW*）、*TF-IDF* 等方法提取文本特征，再结合**朴素贝叶斯（Naive Bayes）**、**逻辑回归（Logistic Regression）**、**支持向量机（SVM）**、**随机森林（Random Forest）**等分类器进行训练和预测。这类方法在特征工程得当的情况下能取得较好效果。

- **基于深度学习的方法**：近年来，以**卷积神经网络（CNN）**、**循环神经网络（RNN 及其变体 LSTM、GRU）**以及 **Transformer 模型（如 BERT、RoBERTa）**为代表的深度学习模型在情感分析任务上取得了显著进展。它们能够自动学习文本的深层语义表示，有效捕捉上下文信息，通常在大型数据集上表现出当前最佳性能，但对计算资源和数据量的要求也相对较高。

本课程报告旨在通过搭建一个统一的实验框架，系统性地实现并对比评估多种**传统机器学习模型**（具体包括逻辑回归、朴素贝叶斯、支持向量机与随机森林）以及一种基础的**前馈神经网络**，在自建的**英文电影评论**数据集上进行情感分类的性能表现，并对实验结果进行初步的分析与讨论。

2 算法理论基础

2.1 文本预处理与 TF-IDF 特征提取

文本预处理是自然语言处理任务中的关键首步，旨在清洗和规范化原始文本数据，以便后续模型能更有效地学习。本报告中采用的预处理流程主要包括：小写转换、去除标点符号、分词（英文文本中基于空格或 NLTK 工具）、以及去除停用词（如 “the”，“is” 等常见但对情感区分贡献不大的词语）。其目的是减少噪声，提取更有意义的文本单元。

在预处理之后，需要将文本转换为数值型特征向量。**TF-IDF（Term Frequency-Inverse Document Frequency，词频-逆文档频率）**是一种广泛应用的文本表示方法。它通过评估一个词语在一个文档中的重要性以及在整个语料库中的普遍性来计算该词的权重。

- **词频（TF）**：指词语 t 在文档 d 中出现的频率，通常进行归一化：

  $$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

  其中 $f_{t,d}$ 是词 t 在文档 d 中的出现次数。

- **逆文档频率（IDF）**：衡量词语 t 的普遍性。如果一个词在很多文档中都出现，其 IDF 值会较低：

  $$IDF(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}| + 1}$$

  其中 $|D|$ 是语料库中的总文档数，$|\{d \in D : t \in d\}|$ 是包含词语 t 的文档数（分母加 1 是为了避免除零错误）。

- **TF-IDF 权重**：词语 t 在文档 d 中的 TF-IDF 权重即为两者的乘积：

  $$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

最终，每篇文档被表示为一个由各词 TF-IDF 权重组成的向量。

2.2 逻辑回归（Logistic Regression）

逻辑回归是一种广泛用于二分类问题的线性分类模型。它通过 **Sigmoid 函数**（亦称 Logistic 函数）将线性组合的输入映射到 (0,1) 区间，表示样本属于某一类别的概率。

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

其中 $z = \mathbf{w}^T\mathbf{x} + b$，$\mathbf{w}$ 是权重向量，$\mathbf{x}$ 是输入特征向量（如 TF-IDF 向量），b 是偏置项。对于情感分类任务（如正面/负面），输出 $P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$ 即为样本 $\mathbf{x}$ 判定为

正面的概率。模型训练的目标是最小化损失函数，常用的如对数损失函数（*Log Loss*）：

$$J(\mathbf{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\sigma(z^{(i)})) + (1-y^{(i)})\log(1-\sigma(z^{(i)}))]$$

特点与局限性：算法简单、计算效率高、易于实现和解释。但它本质上是线性模型，对于非线性可分的数据可能表现不佳。

2.3 朴素贝叶斯（Naive Bayes）

朴素贝叶斯是一类基于**贝叶斯定理**的概率分类算法，其核心思想是假设特征之间条件独立。对于给定的特征向量 $\mathbf{x} = (x_1, \ldots, x_n)$，朴素贝叶斯分类器预测其属于类别 C_k 的后验概率为：

$$P(C_k|\mathbf{x}) = \frac{P(C_k)P(\mathbf{x}|C_k)}{P(\mathbf{x})}$$

由于特征条件独立假设，$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} P(x_i|C_k)$。分类决策规则为选择后验概率最大的类别：

$$\hat{y} = \arg\max_{C_k} P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

在文本分类中常用的是**多项式朴素贝叶斯（MultinomialNB）**，它假设特征（如词频）服从多项式分布。为避免零概率问题，常采用拉普拉斯平滑（*Laplace Smoothing*）：

$$P(w_j|C_k) = \frac{N_{kj} + \alpha}{N_k + \alpha V}$$

其中 N_{kj} 是词 w_j 在类别 C_k 中出现的次数，N_k 是类别 C_k 中的总词数，V 是词汇表大小，α 是平滑参数（通常为 1）。

特点与局限性：算法简单高效，对缺失数据不敏感，在高维数据（如文本）上表现良好，尤其在小数据集上仍有竞争力。其主要局限性在于“朴素”的特征条件独立假设在现实中往往不成立。

2.4 支持向量机（SVM）

支持向量机是一种强大的二分类和多分类模型，其基本思想是在特征空间中找到一个能将不同类别样本分隔开的**最优超平面**，使得该超平面到最近的训练样本点（即支持向量）的距离（间隔，margin）最大化。对于线性可分数据，优化目标为：

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, m$$

对于非线性可分数据，SVM 引入软间隔（*soft margin*）允许一定的错分类，并通过**核技巧（Kernel Trick）**将数据映射到更高维空间使其线性可分。常用核函数包括线性核、多项式核和径向基函数（RBF）核。本报告中主要考虑线性核。

特点与局限性：在高维空间中表现优异，泛化能力强，尤其适用于小样本情况。但训练时间复杂度较高，对参数选择和核函数选择敏感。

2.5 随机森林（Random Forest）

随机森林是一种集成学习算法，它通过构建多个**决策树（Decision Tree）**并综合它们的预测结果来进行分类或回归。其构建过程主要包括：

1. **自助采样（Bootstrap Sampling）**：从原始训练数据集中有放回地随机抽取多个大小相同的子样本集。
2. **特征随机性**：在构建每棵决策树的每个节点时，不是从所有特征中选择最优分裂特征，而是从一个随机选择的特征子集中进行选择。
3. **构建决策树**：为每个子样本集独立构建一棵决策树，通常不进行剪枝。
4. **集成预测**：对于分类任务，最终结果由所有决策树投票决定（多数票原则）。

决策树的分裂标准通常基于信息增益或基尼不纯度（*Gini Impurity*）：

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2$$

其中 p_k 是样本属于类别 k 的概率。

特点与局限性：能够有效处理高维数据，具有较好的抗过拟合能力和噪声鲁棒性，训练速度相对较快。但模型解释性不如单棵决策树。

2.6 前馈神经网络（Feed-forward Neural Network）

前馈神经网络（FNN），亦称多层感知机（MLP），是最基础的人工神经网络模型。数据从输入层单向流经一个或多个隐藏层，最终到达输出层，层间神经元通常是全连接的。每个神经元对其输入进行加权求和，然后通过一个**激活函数**（如 ReLU、Sigmoid、Tanh）产生输出。

$$h_j = f\left(\sum_i w_{ij} x_i + b_j\right)$$

其中 x_i 是输入，w_{ij} 是权重，b_j 是偏置，$f(\cdot)$ 是激活函数，h_j 是隐藏层神经元的输出。对于二元情感分类，输出层通常使用单个神经元和 Sigmoid 激活函数，输出情感为正面的概率。网络通过**反向传播算法（Backpropagation）**和梯度下降法来最小化损失函数（如二元交叉熵）并更新权重。为防止过拟合，常采用 *Dropout* 等正则化技术。

特点与局限性：能够学习复杂的非线性映射关系，具有较强的表示能力。但训练需要较多数据和计算资源，易陷入局部最优，模型解释性较差（“黑箱”模型），且在小型数据集上易过拟合。

2.7 小结

本报告所选用的情感分析算法各具特点：

- **TF-IDF** 为文本提供了有效的数值化特征表示，是后续分类模型的基础。
- **逻辑回归**作为线性模型，实现简单、速度快，是良好的基准模型。
- **朴素贝叶斯**基于概率理论，对文本数据处理高效，尤其在特征维度高时仍有竞争力，但其特征独立假设是其理论局限。
- **支持向量机**在处理高维稀疏数据（如 TF-IDF 特征）时表现出色，泛化能力较强，但对参数和核函数敏感。
- **随机森林**通过集成多个决策树提高了模型的稳定性和准确性，对过拟合不敏感，能处理非线性关系。
- **前馈神经网络**能够学习复杂的非线性模式，具有强大的潜力，但对数据量、计算资源和调参有较高要求，在小数据集上需谨慎使用以防过拟合。

本报告在后续章节将详细介绍这些算法在电影评论情感分析任务中的具体代码实现、实验设置与性能评估结果。

3 代码实践

本节面向动手环节，首先概览项目的目录层级、核心脚本 sentiment_analyzer.py 的主要功能以及所使用的电影评论数据集；随后将按照“环境与依赖准备，数据加载与文本预处理，TF-IDF 特征提取与多模型训练，批量预测、性能评估与结果分析”四个主要步骤，演示完整的复现流程。通过对关键代码片段的逐步说明与解释，读者可以快速在本地环境搭建并运行实验，验证本文所对比分析的多种**机器学习与神经网络情感分类算法**，实现从数据获取、特征工程、模型构建、性能比较到最终交互式应用的端到端实践。**所有代码请参考附录内容**。

3.1 使用方法

1. 首先建议使用虚拟环境，并用命令下载相关的库

```
pip install pandas numpy nltk scikit-learn tensorflow
```

2. 然后，使用如下命令运行项目。每次运行项目，项目会在五个模型重新训练，随机将数据集中的 75% 用于训练，25% 用于测试3.13.23.3

```
python src\sentiment_analyzer.py
```

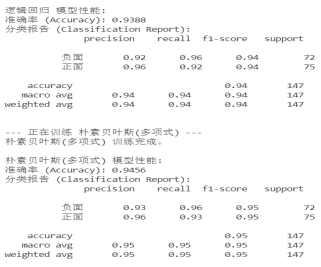

图 3.1: 训练过程 1

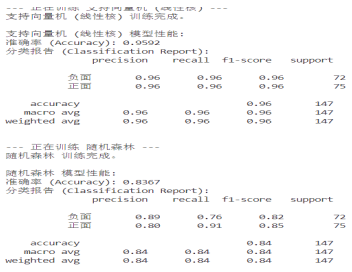

图 3.2: 训练过程 2

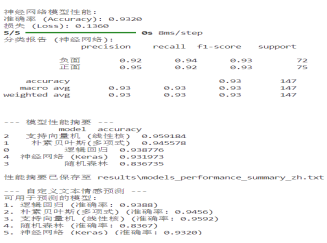

图 3.3: 训练过程 3

3. 最后，你可以自己选择模型，然后输入自己的电影评论，我们会判断是否为恶评或者好评如图：3.4和3.5 所示

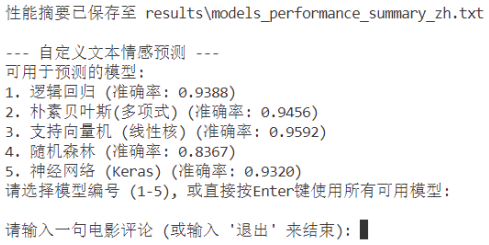

图 3.4: 选择模型

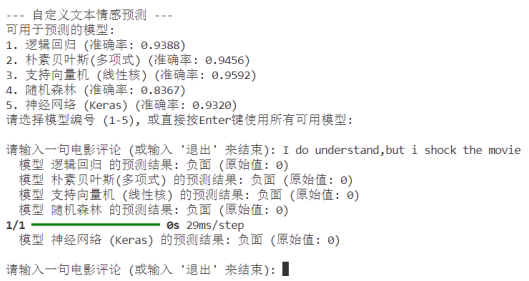

图 3.5: 自主输入结果

3.2 实验结果与分析

3.2.1 实验设置

- **数据集：**采用自建的英文电影评论数据集。原始数据集包含 842 条评论，情感标签包括正面、负面和中性。为进行二元情感分类，实验中首先滤除了 255 条中性评论，剩余 587 条（正面 301 条，负面 286 条）用于后续处理。随后，数据集按照 75% 训练集（440 条）和 25% 测试集（147 条）的比例进行划分，并设置随机种子以保证实验可复现性。
- **文本特征：**对预处理后的文本数据，采用 **TF-IDF（词频-逆文档频率）**进行特征提取，词汇表最大特征数设置为 1000。
- **评估指标：**主要采用准确率（*Accuracy*）、精确率（*Precision*）、召回率（*Recall*）和 F1 值（*F1-score*）作为模型性能的评估标准。对于多类别指标，主要关注宏平均（macro avg）和加权平均（weighted avg）。本次实验主要关注分类效果指标，未对各模型训练与预测速度进行严格计时对比。
- **对比模型：**实验对比了以下五种模型：[label=()]
- **逻辑回归（Logistic Regression）**；
- **朴素贝叶斯（Multinomial Naive Bayes）**；
- **支持向量机（SVM，线性核）**；
- **随机森林（Random Forest）**；
- **前馈神经网络（Feed-forward Neural Network，Keras 实现）**。
- **正则化与参数：**传统机器学习模型主要采用 Scikit-learn 库的默认参数或常用推荐参数(如逻辑回归使用 ‘liblinear‘求解器,随机森林 ‘n_estimators=100‘, ‘max_depth=10‘)。神经网络模型包含两个隐藏层，并使用了 Dropout (0.5) 进行正则化，训练 10 个周期。

3.2.2 各模型性能对比

实验中各模型在测试集上的详细性能指标（精确率、召回率、F1 值针对“负面”和“正面”两类分别计算，并给出宏平均和加权平均）已在运行日志中展示。为便于直观比较，下表总结了各模型的总体准确率以及针对两类情感的 F1 宏平均值。

表 3.1: 各情感分类模型在测试集上的性能对比

模型 (Model)	**准确率 (Accuracy)**	**F1 宏平均 (Macro Avg F1)**
逻辑回归 (Logistic Regression)	0.9388	0.94
朴素贝叶斯 (Multinomial NB)	0.9456	0.95
支持向量机 (SVM, Linear Kernel)	0.9592	0.96
随机森林 (Random Forest)	0.8367	0.84
神经网络 (FNN, Keras)	0.9320	0.93

从表3.1可以看出：

- **支持向量机（线性核）**在所有模型中表现最佳，准确率达到了 0.9592，F1 宏平均值为 0.96。
- **朴素贝叶斯（多项式）**和**逻辑回归**的性能也相当优异，准确率均在 0.93 以上，F1 宏平均值分别达到 0.95 和 0.94。
- **神经网络（Keras）**的准确率为 0.9320，与逻辑回归性能相近，但略低于 SVM 和朴素贝叶斯。
- **随机森林**在此次实验中的表现相对较差，准确率为 0.8367，F1 宏平均值为 0.84。

3.2.3 结果可视化

为了更直观地对比各模型的性能，我们将准确率和 F1 宏平均值进行了可视化处理。图3.8清晰地展示了各模型在准确率（图3.6）和 F1 宏平均值（图3.7）上的对比情况，与表3.1的结论一致。

3.2.4 结果分析

1) 传统机器学习模型的表现 在本次实验中，基于 TF-IDF 特征的传统机器学习模型（SVM、朴素贝叶斯、逻辑回归）均取得了较好的分类效果，准确率都超过了 93%。

- **支持向量机（SVM）**表现最佳。SVM 在高维稀疏数据集（如 TF-IDF 处理后的文本数据）上通常具有良好的泛化能力，其寻找最大间隔超平面的机制有助于实现较好的分类。线性核适用于文本数据，且计算效率相对较高。

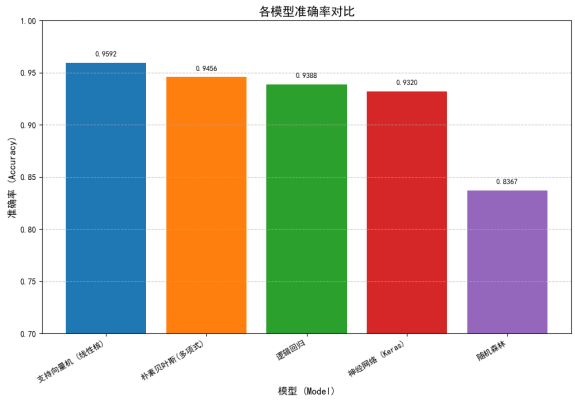

图 3.6: 各模型准确率对比

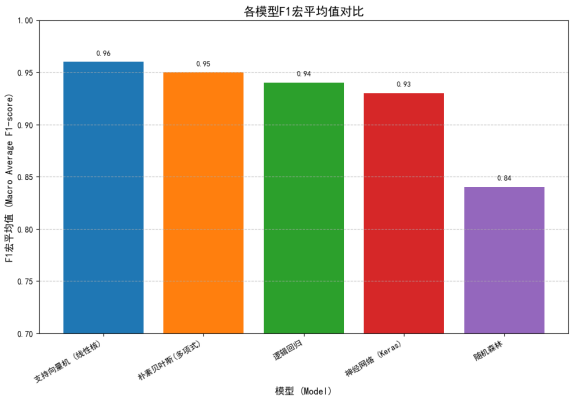

图 3.7: 各模型 F1 宏平均值对比

图 3.8: 各情感分类模型在测试集上的主要性能指标对比

- **朴素贝叶斯**的性能紧随其后。尽管其“朴素”的特征条件独立假设在现实中可能不完全成立，但对于文本分类任务，尤其是当特征维度较高时，朴素贝叶斯往往能以较低的计算成本获得具有竞争力的结果。

- **逻辑回归**作为一种简单而强大的线性模型，也表现出了稳健的性能，是情感分析任务中常用的基线模型。

- **随机森林**的准确率（约 83.7%）明显低于其他三个传统模型。可能的原因包括：(a) 对于当前的 TF-IDF 特征和数据集规模，其默认参数（如树的数量和深度）可能不是最优的；(b) 随机森林中的决策树可能在处理高维稀疏文本特征时不如 SVM 等模型直接有效；(c) 其对“负面”类别召回率较低（0.76），拉低了整体 F1 分数。

2) 神经网络模型的表现与潜力 本次实验中采用的**前馈神经网络(FNN)**获得了 0.9320 的准确率，与逻辑回归性能相当，但略逊于 SVM 和朴素贝叶斯。

- **数据集规模限制：** 正如运行日志中的警告所提示，当前约 587 条（过滤后）用于训练和测试的数据量对于神经网络而言仍然偏小。神经网络通常需要更大量的数据才能充分学习复杂的模式并避免过拟合。实验中，训练集准确率很快达到接近 100%（Epoch 7-9），而验证集准确率在约 93%-94% 波动后最终稳定在 93.2%，这可能表明模型在训练集上存在一定的过拟合，或者在小规模测试集上的泛化能力有限。

- **TF-IDF 特征：** FNN 直接使用 TF-IDF 作为输入特征。虽然 TF-IDF 是有效的文本表示，但它丢失了词序信息。更先进的神经网络模型通常会结合词嵌入（Word Embeddings）或直接处理原始文本序列（如使用 RNN、Transformer）以捕捉更丰富的语义和上下文信息。

- **模型结构与优化：**当前 FNN 结构（两层隐藏层，每层后接 Dropout）相对简单。通过调整网络层数、神经元数量、激活函数、优化器、学习率以及更精细的正则化策略（如早停、L1/L2 正则化），其性能仍有提升空间。

3) 误差来源探讨（共性问题） 尽管部分模型取得了较高的准确率，但在实际情感分析中，仍可能存在以下常见的误差来源：

- **复杂语言现象：**如讽刺、反语、隐喻等，其字面意思与真实情感可能相反或不一致，这对基于词袋模型的 TF-IDF 和大多数分类器都是挑战。
- **否定结构与情感强度：**虽然预处理包含去除停用词，但复杂否定结构（如双重否定）或情感强度的细微差异（如“有点好”vs“非常好”）可能难以准确捕捉。
- **上下文依赖与歧义：**单个词语的情感可能随上下文变化，TF-IDF 无法很好地处理这种动态性。
- **领域特定表达：**电影评论中可能存在的特定俚语或梗，如果未在训练数据中充分出现，模型可能难以理解。

综上 实验结果表明，对于当前规模和特征表示下的英文电影评论二元情感分类任务：

支持向量机（线性核）展现出最优的综合性能；

朴素贝叶斯和**逻辑回归**是非常有效且高效的基线模型；

前馈神经网络在小数据集和 TF-IDF 特征下表现尚可，但其潜力有待进一步挖掘；

随机森林的默认配置可能不适用于当前任务，需要进一步调优。所有模型均在一定程度上成功学习到了情感分类的模式，但仍有提升空间，特别是在处理复杂语言现象和利用更深层语义信息方面。

3.2.5 小结

在本实验中，我们对自建的英文电影评论数据集进行了情感分类。通过统一的预处理和 TF-IDF 特征提取流程，对比了四种传统机器学习模型和一种前馈神经网络的性能。结果显示，**支持向量机（线性核）**在该任务上取得了最佳的准确率（0.9592）和 F1 宏平均值（0.96），紧随其后的是朴素贝叶斯和逻辑回归。神经网络模型表现与逻辑回归相近，但考虑到其对数据量的需求，未来通过增大训练数据、采用词嵌入特征和更复杂的网络结构，其性能有望得到显著提升。随机森林的表现相对不佳，提示了针对特定数据和特征进行模型选择与参数调优的重要性。总体而言，实验验证了这些经典算法在文本情感分析任务上的有效性，并为后续的改进研究提供了基线。

注意： 所有模型的具体超参数设置和代码实现细节请参考附录或项目提交的源代码文件。

4 总结与展望

4.1 工作总结

本文围绕自然语言处理课程项目，设计并实现了一个针对**英文电影评论情感分类**的统一实验框架。该框架系统性地比较了多种经典的机器学习方法和一种基础的神经网络模型。具体而言，集成了基于 **TF-IDF** 文本特征的**逻辑回归（LR）**、**朴素贝叶斯（NB）**、**支持向量机（SVM）**、**随机森林（RF）**四种传统机器学习算法，以及一个**前馈神经网络（FNN）**。工作内容覆盖了数据收集（自建约 1000 条评论数据集）、文本预处理、特征工程、多模型训练与评估（采用准确率、精确率、召回率、F1 值等指标），并实现了一个允许用户交互式输入文本进行情感预测的模块。

4.2 不足与改进空间

尽管本研究搭建了较为完整的实验流程，但在以下方面仍存在不足与可改进之处：

- *数据集规模与多样性*：目前采用的约 1000 条英文电影评论数据集在多样性和规模上仍有限，可能不足以充分发挥深度学习模型的潜力，且其泛化能力至其他类型评论文本（如产品评论、社交媒体文本）有待验证。
- *特征表示的局限性*：主要依赖 TF-IDF 作为文本特征，这种方法忽略了词序信息和深层语义上下文，难以捕捉如讽刺、隐喻等复杂语言现象。
- *模型复杂度与优化*：所采用的前馈神经网络结构相对简单，未进行充分的超参数调优；传统机器学习模型也主要基于默认或经验参数，存在进一步优化的空间。
- *细粒度情感分析的缺失*：当前工作聚焦于文档级的情感极性（正面/负面）判断，未能涉及更细致的方面级情感分析（如对电影的演员、剧情、画面等不同方面的情感）或情感强度、具体情绪类别的识别。

4.3 未来启发探究

基于当前工作的基础，未来可以从以下几个方面进行更深入的探索与研究：

- **高级特征表示**：引入**词嵌入（如 Word2Vec, GloVe, FastText）**或**预训练语言模型（如 BERT, RoBERTa）**的上下文向量表示，以捕获更丰富的语义信息和上下文依赖关系，提升模型对文本的理解深度。
- **复杂模型架构**：探索并应用更先进的深度学习模型，如**循环神经网络（RNN/LSTM/GRU）**、**卷积神经网络（CNN）**以及 **Transformer** 架构，这些模型在序列数据处理和特征提取方面具有更强能力。

- **细粒度情感分析**：扩展研究至**方面级情感分析（ABSA）**，识别文本中针对特定实体或属性的情感倾向；或进行**情感强度分析**与**多情绪分类**，实现更全面的情感理解。

- **模型鲁棒性与可解释性**：研究模型在面对俚语、拼写错误、对抗性攻击等噪声数据时的鲁棒性，并采用如 **LIME**、**SHAP** 等技术分析模型决策依据，增强模型的可解释性与可信度。

- **跨领域与小样本学习**：研究如何利用迁移学习、领域自适应等技术，将现有模型有效应用于数据稀缺或特征分布不同的新领域；探索小样本学习方法以减少对大规模标注数据的依赖。

4.4 结语

本文通过设计和实现一个包含数据处理、多种基线模型训练、评估及交互式预测的**电影评论情感分析框架**，为理解不同算法在该特定 NLP 任务上的特性与表现提供了一个可复现且可扩展的实验平台。通过对所用方法、数据集构建过程的阐述以及对实验结果的初步分析，本工作为后续在**语义表示学习**、**复杂模型应用**、**细粒度情感挖掘以及模型泛化能力提升**等方向的深入研究奠定了基础。期望本报告的实践与总结能为自然语言处理领域的学习者和研究者提供有益的参考，并激发对情感分析技术进一步探索的热情。

参考文献

[1] Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval*, 2(1–2), 1–135.

[2] Salton, G., & Buckley, C. (1988). *Term-weighting approaches in automatic text retrieval. Information Processing & Management*, 24(5), 513–523.

[3] McCallum, A., & Nigam, K. (1998). *A comparison of event models for naive Bayes text classification.* In *AAAI-98 Workshop on Learning for Text Categorization*, 752(1), 41–48.

[4] Cortes, C., & Vapnik, V. (1995). *Support-vector networks. Machine Learning*, 20(3), 273–297.

[5] Breiman, L. (2001). *Random forests. Machine Learning*, 45(1), 5–32.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.* (2011). *Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research*, 12, 2825–2830. (Relevant as the project uses Scikit-learn for implementing traditional ML models).

[7] Maas, A. L., Daly, R. E., Pham, P. T., *et al.* (2011). *Learning word vectors for sentiment analysis.* In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. (Introduced a widely used movie review dataset for sentiment analysis and explored word vectors).

[8] Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing.* Morgan & Claypool Publishers. (Provides a good overview of NNs in NLP, relevant to the FNN used).

[9] Jurafsky, D., & Martin, J. H. (2023). *Speech and Language Processing* (3rd ed.). Prentice Hall. (Standard NLP textbook covering text classification, sentiment analysis, and many of the base concepts).

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer. (Covers many machine learning algorithms in depth, including Logistic Regression and SVMs).

A 附录 I: 代码实现

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import re
import string
import os

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

try:
    stopwords.words('english')
    word_tokenize("test")
except LookupError:
    print("NLTK资源未找到。正在下载...")
    nltk.download('stopwords', quiet=True)
    nltk.download('punkt', quiet=True)
    print("NLTK资源下载完成。")

def load_data(filepath):
    print(f"正在从 {filepath} 加载数据...")
    if not os.path.exists(filepath):
        print(f"错误: 在 {filepath} 未找到数据文件")
        return None
```

```
35      df = pd.read_csv(filepath)
36      print("数据加载成功。")
37      print("原始数据预览:")
38      print(df.head())
39      print(f"\n原始数据集形状: {df.shape}")
40      print(f"原始情感分布:\n{df['sentiment'].value_counts()}")
41      return df
42
43  def preprocess_text(text):
44      if not isinstance(text, str):
45          return ""
46      text = text.lower()
47      text = re.sub(r'<[^>]+>', '', text)
48      text = text.translate(str.maketrans('', '', string.punctuation))
49      tokens = word_tokenize(text)
50      stop_words_set = set(stopwords.words('english'))
51      filtered_tokens = [word for word in tokens if word.isalnum() and
            word not in stop_words_set]
52      return " ".join(filtered_tokens)
53
54  def train_and_evaluate_sklearn_model(model, X_train, y_train,
        X_test, y_test, model_name, target_names):
55      print(f"\n--- 正在训练 {model_name} ---")
56      model.fit(X_train, y_train)
57      print(f"{model_name} 训练完成。")
58
59      y_pred = model.predict(X_test)
60      accuracy = accuracy_score(y_test, y_pred)
61      report = classification_report(y_test, y_pred, target_names=
            target_names, zero_division=0)
62
63      print(f"\n{model_name} 模型性能:")
64      print(f"准确率 (Accuracy): {accuracy:.4f}")
65      print("分类报告 (Classification Report):")
66      print(report)
67      return {'model': model_name, 'accuracy': accuracy, 'report': report, '
```

```
        'trained_model': model}

def create_nn_model(input_dim):
    model = Sequential([
        Dense(128, activation='relu', input_dim=input_dim),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model

def train_and_evaluate_nn_model(X_train_tfidf, y_train, X_test_tfidf, y_test, target_names):
    print("\n--- 正在训练神经网络 ---")

    X_train_dense = X_train_tfidf.toarray()
    X_test_dense = X_test_tfidf.toarray()

    input_dim = X_train_dense.shape[1]
    nn_model = create_nn_model(input_dim)

    print("神经网络模型摘要:")
    nn_model.summary()

    y_train_nn = np.asarray(y_train).astype('float32')
    y_test_nn = np.asarray(y_test).astype('float32')

    history = nn_model.fit(X_train_dense, y_train_nn,
                           epochs=10,
                           batch_size=16,
                           validation_data=(X_test_dense, y_test_nn),
                           verbose=1)
```

```
100
101         loss, accuracy = nn_model.evaluate(X_test_dense, y_test_nn,
                verbose=0)
102         print(f"\n神经网络模型性能:")
103         print(f"准确率 (Accuracy): {accuracy:.4f}")
104         print(f"损失 (Loss): {loss:.4f}")
105
106         y_pred_proba = nn_model.predict(X_test_dense)
107         y_pred_nn = (y_pred_proba > 0.5).astype(int).flatten()
108
109         report_nn = classification_report(y_test_nn, y_pred_nn,
                target_names=target_names, zero_division=0)
110         print("分类报告 (神经网络):")
111         print(report_nn)
112
113         return {'model': '神经网络 (Keras)', 'accuracy': accuracy, 'report':
                report_nn, 'trained_model': nn_model}
114
115 def predict_user_input(models_results, tfidf_vectorizer,
        target_names_map):
116         print("\n--- 自定义文本情感预测 ---")
117
118         if not models_results:
119             print("没有可用于预测的已训练模型。")
120             return
121
122         print("可用于预测的模型:")
123         for i, res in enumerate(models_results):
124             print(f"{i + 1}. {res['model']} (准确率: {res['accuracy']:.4f})")
125
126         model_choice_idx = -1
127         while True:
128             try:
129                 choice_str = input(f"请选择模型编号 (1-{len(models_results)}),
                        或直接按Enter键使用所有可用模型: ")
130                 if not choice_str:
```

```
131                 model_choice_idx = None
132                 break
133             model_choice_idx = int(choice_str) - 1
134             if 0 <= model_choice_idx < len(models_results):
135                 break
136             else:
137                 print(f"请输入1到{len(models_results)}之间的数字。")
138         except ValueError:
139             print("无效输入。请输入一个数字。")
140
141     while True:
142         user_text = input("\n请输入一句电影评论 (或输入 '退出' 来结束): ")
143         if user_text.lower() == '退出':
144             break
145         if not user_text.strip():
146             print("请输入一些文本。")
147             continue
148
149         preprocessed_text = preprocess_text(user_text)
150         transformed_text_tfidf = tfidf_vectorizer.transform([
                preprocessed_text])
151
152         models_to_predict_with = []
153         if model_choice_idx is not None:
154             models_to_predict_with.append(models_results[
                    model_choice_idx])
155         else:
156             models_to_predict_with = models_results
157
158         for res in models_to_predict_with:
159             current_model = res['trained_model']
160             model_name = res['model']
161
162             prediction = None
163             predicted_label = None
164
```

```
165             if model_name == '神经网络 (Keras)':
166                 transformed_text_dense = transformed_text_tfidf.
                        toarray()
167                 pred_proba = current_model.predict(
                        transformed_text_dense)
168                 prediction = (pred_proba > 0.5).astype(int).flatten()[0]
169             else:
170                 prediction = current_model.predict(
                        transformed_text_tfidf)[0]
171
172             predicted_label = target_names_map.get(prediction, "未知
                    ")
173             print(f"  模型 {model_name} 的预测结果: {predicted_label} (原
                    始值: {prediction})")
174
175 def main():
176     data_filepath = "data/movie_reviews.csv"
177
178     df = load_data(data_filepath)
179     if df is None:
180         return
181
182     print("\n正在过滤 'neutral' (中性) 评论以进行二元分类 (正面/负面)...")
183     df = df[df['sentiment'] != 'neutral'].copy()
184     if df.empty or len(df['sentiment'].unique()) < 2:
185         print("错误: 过滤 'neutral' 评论后数据不足或类别少于2。请确保CSV
                文件中有 'positive' 或 'negative' 情感。")
186         return
187     print("'neutral' 评论过滤后的情感分布:")
188     print(df['sentiment'].value_counts())
189
190     y = df['sentiment'].apply(lambda s: 1 if s == 'positive' else 0)
191     target_names_for_report = ['负面', '正面']
192     target_names_map = {0: '负面', 1: '正面'}
193
194     if 'review' not in df.columns:
```

```
195         print("错误: 数据集必须包含 'review' 列。")
196         return
197
198     print("\n正在预处理文本数据...")
199     df['cleaned_review'] = df['review'].astype(str).apply(preprocess_text)
200     print("文本预处理完成。")
201
202     X = df['cleaned_review']
203
204     if X.shape[0] < 2:
205         print("错误: 数据不足，无法继续模型训练。")
206         return
207
208     try:
209         X_train, X_test, y_train, y_test = train_test_split(X, y,
                test_size=0.25, random_state=42, stratify=y)
210     except ValueError:
211         print("警告: 分层抽样失败，可能是因为某个类别的样本过少。将不使
                用分层进行划分。")
212         X_train, X_test, y_train, y_test = train_test_split(X, y,
                test_size=0.25, random_state=42)
213
214     if X_train.empty or X_test.empty:
215         print("错误: 划分后训练集或测试集为空。")
216         return
217
218     print(f"\n训练数据形状: X_train: {X_train.shape}, y_train: {y_train.
            shape}")
219     print(f"测试数据形状: X_test: {X_test.shape}, y_test: {y_test.shape}")
220
221     print("\n正在提取TF-IDF特征...")
222     tfidf_vectorizer = TfidfVectorizer(max_features=1000)
223     X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
224     X_test_tfidf = tfidf_vectorizer.transform(X_test)
225     print("TF-IDF特征提取完成。")
226
```

```
227     sklearn_models = {
228         "逻辑回归": LogisticRegression(random_state=42, solver='
                liblinear', C=1.0),
229         "朴素贝叶斯(多项式)": MultinomialNB(alpha=1.0),
230         "支持向量机 (线性核)": SVC(kernel='linear', probability=True,
                random_state=42, C=1.0),
231         "随机森林": RandomForestClassifier(n_estimators=100,
                random_state=42, max_depth=10)
232     }
233
234     all_models_results = []
235
236     for name, model in sklearn_models.items():
237         results = train_and_evaluate_sklearn_model(model,
                X_train_tfidf, y_train, X_test_tfidf, y_test, name,
                target_names_for_report)
238         all_models_results.append(results)
239
240     print("\n警告: 当前数据集对于神经网络来说较小，结果可能不是最优的。
            ")
241     try:
242         nn_results = train_and_evaluate_nn_model(X_train_tfidf,
                y_train, X_test_tfidf, y_test, target_names_for_report)
243         all_models_results.append(nn_results)
244     except Exception as e:
245         print(f"训练或评估神经网络时出错: {e}")
246         print("将跳过神经网络模型。")
247
248     print("\n\n--- 模型性能摘要 ---")
249     results_summary_df = pd.DataFrame(all_models_results,
            columns=['model', 'accuracy'])
250     print(results_summary_df.sort_values(by='accuracy', ascending=
            False))
251
252     results_dir = "results"
253     if not os.path.exists(results_dir):
```

```
254         os.makedirs(results_dir)
255     summary_filepath = os.path.join(results_dir, "
            models_performance_summary_zh.txt")
256     with open(summary_filepath, "w", encoding="utf-8") as f:
257         f.write("模型性能摘要:\n")
258         f.write(results_summary_df.sort_values(by='accuracy',
                ascending=False).to_string(index=False))
259     print(f"\n性能摘要已保存至 {summary_filepath}")
260
261     predict_user_input(all_models_results, tfidf_vectorizer,
            target_names_map)
262
263     print("\n分析完成。")
264
265 if __name__ == '__main__':
266     physical_devices = tf.config.list_physical_devices('GPU')
267     if physical_devices:
268         try:
269             tf.config.experimental.set_memory_growth(
                    physical_devices[0], True)
270         except RuntimeError as e:
271             print(e)
272     main()
```