1. 实验简介

1.1 实验目标

本实验的主要目标如下：

1) 实现目标：基于简化版 Transformer 编码器，构建一个中文文本情感分类模型，实现对评论文本的情感极性（正/负）识别。
2) 训练目标：利用训练集对模型进行端到端训练，学习词语间的上下文语义特征，提升模型对情绪倾向的识别能力。
3) 评估目标：在验证集与测试集上对模型进行评估，计算准确率（Accuracy）、损失值（Loss）等指标，判断模型的泛化能力与实际性能。
4) 应用价值：本实验训练得到的模型可广泛应用于舆情监控、用户反馈分析、推荐系统优化等多个现实场景中，为企业和研究机构提供智能化的情感洞察支持。

1.2 实验环境与设置

为验证基于 Transformer 的文本分类模型在中文情感分析任务中的有效性，本文在标准的数据集与硬件环境下开展实验，实验设置包括数据集准备、运行环境配置及超参数设置等内容。

1.2.1 数据集

本实验使用公开的中文情感分析语料库 ChnSentiCorp，其包含来自电商评论、电影评论等多个领域的真实中文评论文本。数据集已划分为训练集（train.tsv）、验证集（dev.tsv）与测试集（test.tsv），每条样本由情感标签与评论文本组成，格式如下：

表 1 数据集格式

label	text
0	这个电影真的很好看！
1	剧情无聊，太失望了。

- 标签解释：0 表示消极情绪，1 表示积极情绪；
- 数据规模：训练集 9600 条，验证集 1200 条，测试集 1200 条；
- 文本预处理：实验中采用简化的分词器进行字符级编码，并进行统一长度截断或填充（最大长度为 128）。

1.2.2 实验环境

实验在高性能 GPU 服务器上进行，具体软硬件配置如下：

GPU：NVIDIA A100 40GB，驱动版本 520.61.07，CUDA 版本 11.8；
操作系统：Linux（Ubuntu 20.04）；
Python 版本：3.10；
主要依赖库：

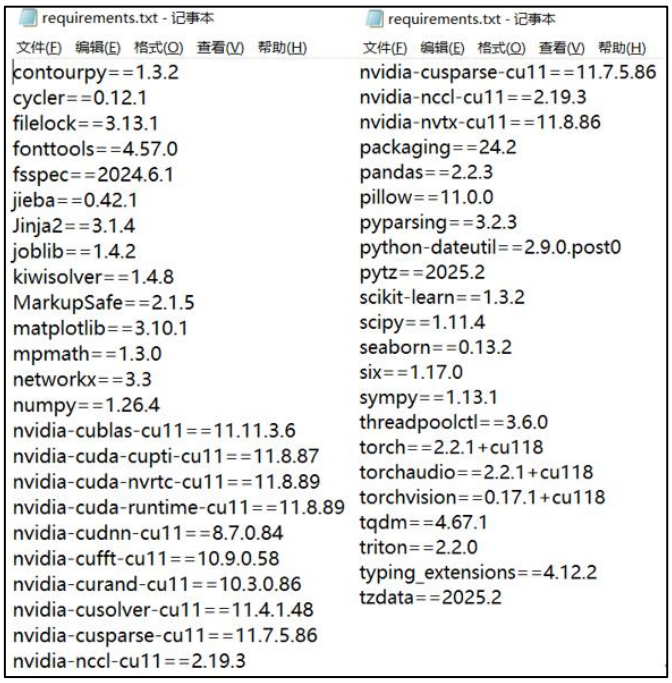

图 1 环境依赖的库

1.2.3 超参数设置

本实验所使用的关键模型超参数如下表所示：

表 2 模型超参数值

参数	数量
批次大小（batch_size）	32
学习率（learning_rate）	2.00E-04
训练轮数（num_epochs）	20
最大文本长度（max_len）	128
损失函数	交叉熵损失（CrossEntropy）

此外，模型各部分的参数量如下：

- 词嵌入层：384,000 个参数；
- Transformer 编码器层（2 层，每层参数量 527,104）：共 1,054,208 个参数；
- 分类全连接层：514 个参数；
- 总参数量：649,218。

这些设置保证了模型在训练期间具有较强的表达能力，同时在合理资源约束下保持较高的训练效率与稳定性。

2. 模型设计

本实验采用简化版 Transformer 编码器模型来完成中文文本的二分类任务，整体结构简洁、可控，适合中小规模数据场景。模型的设计灵感来源于原始 Transformer 模型，但移除了复杂的解码器结构，仅保留编码器部分用于文本表示学习。

2.1 模型架构

本实验模型主要由以下几层组成：

（1）输入层

输入为一段经过分词的中文文本序列，首先通过词嵌入层将离散的词 ID 转换为连续的稠密向量，形成维度为$batch_size \times seq_len \times embedding_dim$的张量。

（2）位置编码层

由于 Transformer 本身不具备序列建模能力，因此需显式引入位置信息。本模型采用可学习位置编码，即为每个位置引入独立的向量，与词向量相加作为最终的输入表示。

（3）Transformer 编码器

该部分由 $N = 2$ 个 Transformer Encoder Layer 堆叠组成，每层包含以下子模块：

• 多头自注意力机制

输入序列中每个位置的词向量都能关注序列中其他位置，通过加权方式动态聚合上下文信息。使用多头机制可在不同子空间中并行建模不同语义关系。

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

• 前馈全连接网络

每个位置的表示经过两层全连接层，进行非线性变换。第一层用于升维，第二层降回原维度，搭配 ReLU 激活函数。

• 残差连接与层归一化

每个子模块外接残差连接和 LayerNorm，避免梯度消失、提升训练稳定性。

（4）分类层

Transformer 输出的序列向量中，取第一个位置的表示（类似于 BERT 中的 [CLS]），输入至全连接层进行分类，输出 logits 分数，并使用交叉熵损失函数进行训练。

2.2 模型参数设置

表 3 列出了模型各项关键超参数配置。

表 3 模型各项关键超参数配置

参数	数量	说明
词汇表大小（vocab_size）	3000	数据集中独立词汇数量
嵌入维度（embedding_dim）	128	每个词的向量维度
隐藏层维度（hidden_dim）	256	Transformer 层中 Q/K/V 及 FFN 中间维度
注意力头数（num_heads）	4	每层多头注意力机制的头数
编码器层数（num_layers）	2	Transformer Encoder 堆叠层数
最大序列长度（max_len）	128	输入文本的最大长度，超过部分截断
类别数（num_classes）	2	二分类任务
Dropout 概率	0.1	防止过拟合

2.3 模型图

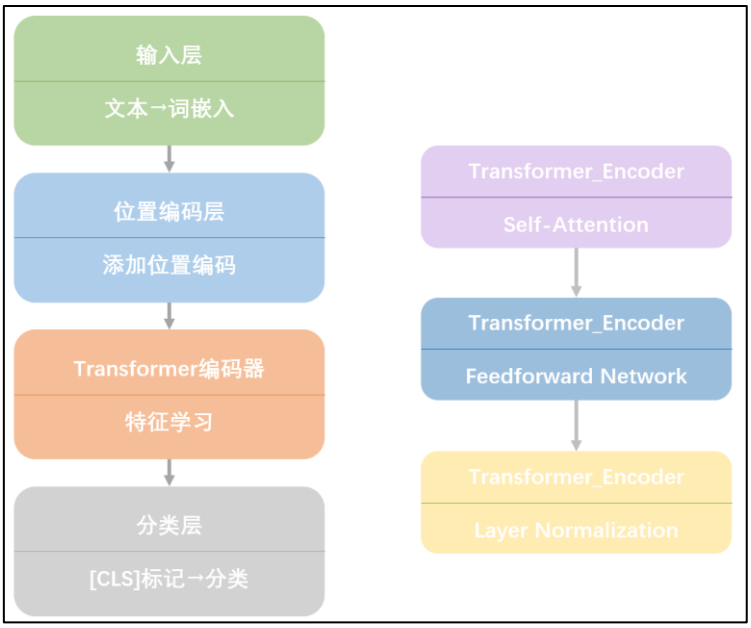

图 2 模型结构

3. 实验过程与结果分析

3.1 实验流程

本实验主要流程包括数据预处理、模型训练、模型评估三个阶段，如下所示：

1) 数据加载与预处理

读取 train.tsv、dev.tsv 和 test.tsv 三个数据文件，并对中文文本进行清洗与分词处理。通过自定义的 SimpleTokenizer 构建词汇表并将文本编码为对应的索引序列，同时使用固定的最大长度（128）进行截断与填充，确保输入维度一致。

2) 模型训练

使用训练集对基于 Transformer 的文本分类模型进行训练。采用交叉熵作为损失函数，并使用 Adam 优化器进行权重更新。每轮训练结束后在验证集上评估模型的准确率，并保存验证准确率最优的模型参数。

3) 模型评估

在训练完成后，使用保存的最优模型在测试集上进行性能评估，计算准确率、精确率、召回率和 F1 分数等评价指标，并绘制混淆矩阵。

3.2 模型训练与验证结果

在 20 个训练周期中，训练损失不断下降，验证集准确率稳定提升，说明模型在训练过程中逐步收敛，具备良好的泛化能力。

- 训练集平均损失：0.0389
- 验证集准确率：86.67%
- 测试集准确率：87.08%

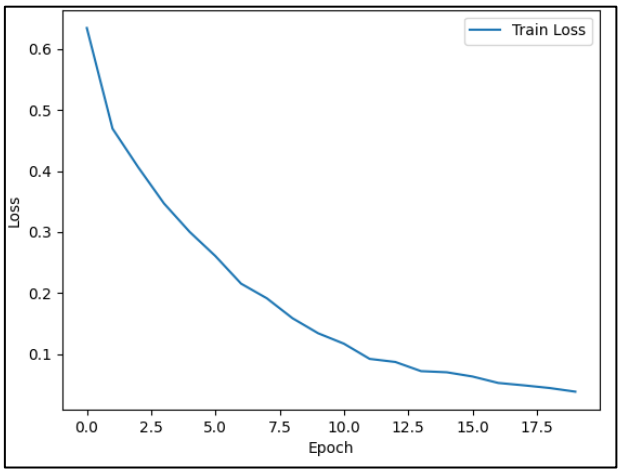

图 3 模型训练损失曲线

通过 matplotlib 绘制的损失曲线显示，训练初期损失下降较快，后期逐渐趋于平稳，验证了模型的稳定训练过程。

3.3 测试结果与性能评估

在测试集上的表现如下：

- 测试集准确率：87.08%
- 混淆矩阵（基于 seaborn 绘制）如下：

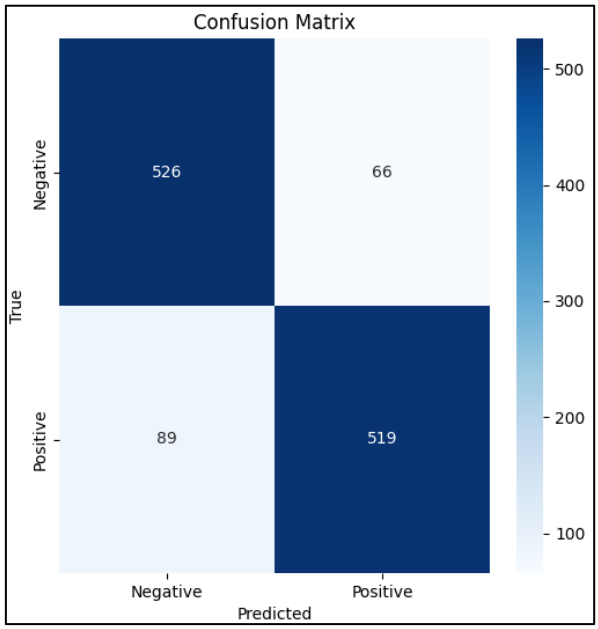

图 4 混淆矩阵热力图

表 4 混淆矩阵

实际类 \ 预测类	正类（1）	负类（0）
正类（1）	519	89
负类（0）	66	526

根据混淆矩阵计算模型主要性能指标如下：

- 准确率（Accuracy）：

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{526 + 519}{526 + 66 + 89 + 519} = \frac{1045}{1200} \approx 0.8708 \quad (2)$$

• 精确率（Precision）：

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{519}{519 + 66} = \frac{519}{585} \approx 0.8875 \quad (3)$$

• 召回率（Recall）：

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{519}{519 + 89} = \frac{519}{608} \approx 0.8539 \quad (4)$$

• F1 分数（F1 Score）：

$$\text{F1Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \times \frac{0.8875 \times 0.8539}{0.8875 + 0.8539} \approx 0.8702 \quad (5)$$

这些指标表明模型在二分类任务中具有良好的综合性能。较高的精确率说明模型对正类的预测可靠，而较高的召回率说明模型能够有效识别大多数正类样本。

3.4 错误分析

尽管模型整体性能较好，但仍存在一些误分类问题：

• 假阳性（FP）：将 66 条负类误判为正类，可能是由于某些负面评论中包含积极词汇，导致模型产生混淆；

• 假阴性（FN）：将 89 条正类误判为负类，可能是由于评论语言隐晦、带有讽刺，或输入文本过短导致特征提取不足。